From genes to organisms: Bioinformatics System Models and Software

THÈSE N^{O} 6081 (2014)

PRÉSENTÉE LE 21 FÉVRIER 2014

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE SYSTÈMES INTELLIGENTS
PROGRAMME DOCTORAL EN BIOTECHNOLOGIE ET GÉNIE BIOLOGIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Thomas SCHAFFTER

acceptée sur proposition du jury:

Prof. B. Deplancke, président du jury
Prof. D. Floreano, Prof. M. Affolter, directeurs de thèse
Prof. E. Hafen, rapporteur
Dr J. J. Rice, rapporteur
Prof. M. Unser, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2014

Acknowledgements

Although presented as a personal contribution to science, this thesis would not have been possible without the collaboration and support of many people.

First of all, I would like to thank my thesis directors Prof. Dario Floreano and Prof. Markus Affolter. Before joining the Laboratory of Intelligent Systems (LIS), I had specialized in robotics and artificial intelligence for my Master degree. Dario and his collaborators made me discover another exciting field, computational biology. I wish to express my gratitude to Dario for providing me with the freedom to develop my research, and for giving me his support and trust. Moreover, he gave me precious advices on how to organize and expose scientific ideas, which is invaluable for my professional experience. I also want to thank Markus for having accepted to co-supervise this thesis and for sharing with me his expertise and never ending enthusiasm for molecular and developmental biology.

I would like to thank Prof. Bart Deplancke, Prof. Michael Unser, Prof. Ernst Hafen, and Dr John Jeremy Rice for having accepted to be members of my thesis committee, and for their time and effort in reviewing this work. Thanks are also due to the Swiss Initiative in Systems Biology SystemsX.ch for funding the WingX project and my research. I particularly value the unique opportunity to meet many researchers from different backgrounds (biology, mathematics, physics, computer sciences, etc.) and research groups all over Switzerland including ETH Zurich, Universities of Zürich, Bern and Basel, and EPFL.

An important contribution of my PhD is the development and implementation of many methods and computational tools. This gave me a unique opportunity to learn from many research fields including computational biology, artificial intelligence, optimization algorithms, graph and network theory, image analysis, information visualization and human-computer interaction. I was also able to develop extensible and user-friendly computational tools, which are available today to a broad community of biologists and computer scientists. However, this performance would not have been possible without the contribution of few people. I would like to thank Dr Daniel Marbach for making me discover the world of computational biology and for his collaboration on *GeneNetWeaver* and the DREAM project; Dr Ricard Delgado-Gonzalo for his expertise on active contour segmentation (also known as *snakes*) and for his fruitful collaboration on *WingJ*; Dr Fisun Hamaratoglu for having accepted to follow me on the WingJ project and for providing us with hundreds of confocal fluorescence images of the wings of

fruit flies (*Drosophila melanogaster*); and Dr Aitana Neves da Silva for answering countless questions about the development of the *Drosophila* embryo.

My other colleagues at the Laboratory of Intelligent Systems also deserve my gratitude. I would like to thank Dario once again for creating such a friendly and stimulating atmosphere at his lab, and for entrusting me with the role of system administrator. Moreover, I want to thank the members of the *Evolutionary group* for constructive discussions: Dr Daniel Marbach, Dr Peter Dürr, Dr Claudio Mattiussi, Dr Pradeep Fernando, Andrea Maesani and Trevis Alleyne. To all the others, thank you for the great discussions, cooking battles, movie nights, evenings at Sat and, more generally for the great time we had!

I would like to express my sincere gratitude to Michelle Wälti, Anouk Hein, and Sonja Bodmer for their availability and efficient work in helping me out with administrative matters. I also want to thank Claude Waeber from the Audiovisual group for assisting me with the setting of the multi-point video conferences with Zürich, Bern, and Basel. Also thanks to EPFL to who I own most of my scientific and engineering education, and for providing me with a dynamic and creative environment.

I take this opportunity to thank all my friends who supported me through all these years. A special thanks goes to Yannick Weibel and Severin Leven for making sure I had a social life besides my PhD, and Léda Gerber for the great time. To all the others, I don't cite your names but thank you for the unforgettable memories.

Finally, this work would not have been possible without the support of my family. Thank you to my parents Danièle and Vital who always encouraged me in my work and provided a nurturing environment. Thank you also to my brothers Clément and Gautier for their unconditional support through all these years.

Lausanne, December 2013

Abstract

The expression of genes is controlled by regulatory networks, which perform specific functions in a cell. Gene networks play a crucial role in the development of multicellular organisms by precisely coordinating spatial and temporal gene expression patterns during different developmental stages. Unravelling and modelling these networks is of key importance to gain eventually a complete understanding of developmental processes and genetically related diseases. In this thesis, we present a comprehensive framework for reverse engineering gene regulatory networks, which required the development of many methods in very diverse research fields. A second important contribution is their implementation as extensible, user-friendly and open source computational tools[a].

Over the last decade, numerous methods have been developed for inference of regulatory networks from gene expression data. However, relatively little effort has been put into evaluating the performance of those methods due to the difficulty of constructing adequate benchmarks and the lack of tools for a differentiated analysis of network predictions on such benchmarks. Here, we describe a novel and comprehensive method for *in silico* benchmark generation and performance profiling of network inference methods available to the community as an open-source software called *GeneNetWeaver (GNW)*. In addition to the generation of detailed dynamical models of gene regulatory networks to be used as benchmarks, GNW provides a network motif analysis that reveals systematic prediction errors, thereby indicating potential ways of improving inference methods. The accuracy of network inference methods is evaluated using standard metrics such as precision-recall and receiver operating characteristic (ROC) curves. Furthermore, we used GNW to provide the international DREAM (Dialogue for Reverse Engineering Assessments and Methods) competition with three network inference challenges (DREAM3, DREAM4, and DREAM5). In the context of the DREAM competition, 91 teams submitted about 900 network predictions to evaluate the performance of their methods on GNW-generated benchmarks. Today, the accuracy of more than 25,000 gene network reconstructions have been evaluated by GNW users.

Gene regulatory networks are often organized into groups, modules or community of related genes and proteins carrying out specific biological functions. Here, we also address the rational decomposition of (reconstructed) biological networks into function modules. We present

[a] tschaffter.ch

an extensible and modular framework for community structure detection in networks called *Jmod*. Jmod implements state-of-the-art community structure detection methods including Newman's spectral algorithm and a genetic algorithm-based modularity optimization method that we developed. The performance of these methods has been evaluated on biological and *in silico* networks. The application of these methods is actually not limited to gene regulatory networks as they can also provide insight into the community structure of neural, social, and technological networks, for instance. However, modularity optimization methods are known to be affected by a resolution limit that makes them fail to detect small communities in large networks. Although several attempts have been proposed to overcome this limitation of modularity based methods, none of them solves it in a satisfactorily manner. Therefore, a community voting method was developed and implemented for combining multiple partitions obtained using our GA-based method into one partition more robust and reliable than the individual partitions. We have shown that this approach successfully overcome the resolution limit. Furthermore, our method is best performer along with another method in a comparative analysis that profiled the performance of twelve state-of-the-art community structure detection algorithms.

The reconstruction of a developmental gene network in its spatial context remains a considerable challenge. One of the reason is that this process requires tremendous amount of spatial and temporal gene expression data, which are usually available in very limited quantities due to the inherent difficulty in measuring gene expression in an entire organism. Another contribution of this thesis is the development of an image processing application named *WingJ* for unsupervised and systematic quantification of the developing *Drosophila* wing, which is a classical model for studying the genetic control of tissue size, shape and patterning. First, a parametric model of the morphology or structure of the *Drosophila* wing is inferred from fluorescence images. The segmentation method is based on the design of multiple image processing detection modules, each focusing on the extraction of a specific feature of the wing structure including its orientation. The approach was later extended to the detection of the *Drosophila* embryo. The inferred structure model was then used as a convenient coordinate system for measuring gene and protein expression levels. An important feature of the obtained expression maps is that they can be used to compare domains of expression in differentiated systems, for example to visualize the difference in patterns of gene activity between wild type and mutant wings or in wings imaged at different time points during development. Moreover, a robust, multiscale quantitative description of the developing wing is obtained by combining morphological and gene expression information from multiple wings, completed by the output of an automatic cell nuclei detection method that we have developed. We have used the above method to automatically generate robust quantitative descriptions of wild-type and mutant (*pent* deficient) *Drosophila* wings imaged at 80, 90, 100, and 110 hours after egg laying. Furthermore, we have shown that these quantitative descriptions can be used to unravel the regulatory interactions of a six-gene wing developmental network.

Keywords: gene networks, community detection, data integration, reverse engineering, *Drosophila* wing, image segmentation, multiscale models, computational tools

Résumé

L'expression des gènes résulte de leur organisation en réseaux ce qui permet l'exécution de tâches bien définies dans la cellule. Les réseaux de gènes jouent un rôle crucial dans le développement des organismes multicellulaires en contrôlant précisément le domaine d'expression des gènes dans l'espace et au court du temps. L'identification et la modélisation de ces réseaux est de première importance pour améliorer la compréhension des processus de développement cellulaire et des maladies génétiques. Dans cette thèse, nous présentons un environnement permettant la reconstruction de ces réseaux. Cet environnement a nécessité le développement d'un nombre important de méthodes dans des domaines de recherche très variés. Une autre contribution importante est leur implémentation en une suite de logiciels[b] extensibles et facile d'utilisation.

Un grand nombre de méthodes ont été développées pour reconstruire des réseaux de gènes à partir de leur niveau d'expression. Cependant, des efforts relativement faibles ont été consentis pour les évaluer, ce qui est principalement dû à la difficulté de générer des scénarios de test réalistes et du manque d'outils pour évaluer leur performance sur de tels benchmarks. Nous décrivons ici une nouvelle approche de génération de scenarios de test *in silico* et l'évaluation des performances. Nous avons implémenté cette solution dans un logiciel nommé *GeneNetWeaver (GNW)*. En plus de la génération de réseaux de gènes « virtuels » pour l'évaluation des méthodes, GNW fournit un outil pour une analyse de motifs de réseaux mettant en évidence les erreurs systématiques inhérentes à la méthode, donnant ainsi des indications pour son amélioration. La qualité de la reconstruction est également évaluée en utilisant des mesures standards telles que les courbes «precision-recall» ou ROC. Par ailleurs, nous avons utilisé GNW pour générer des réseaux de gènes virtuels pour la compétition internationale DREAM (Dialog for Reverse Engineering Assessments and Methods) trois années consécutives (DREAM3, DREAM4 et DREAM5). A cette occasion, 91 équipes ont généré 900 prédictions de réseaux de gènes que nous avons construits en utilisant GNW. Aujourd'hui, plus de 25'000 reconstructions de réseaux de gènes qui ont été évaluées par des utilisateurs de GNW.

Les réseaux de gènes sont généralement organisés en groupes, modules ou communautés de gènes réalisant des fonctions biologiques bien spécifiques. Une autre contribution de cette thèse est l'identification de ces modules dans des réseaux de gènes (précédemment

[b] tschaffter.ch

reconstruits). Une méthode extensible et modulaire est présentée pour détecter la structure de communautés de ces réseaux. Cette méthode est inclue dans un logiciel appelé *Jmod* qui implémente quelques-uns des meilleurs algorithmes tels que la méthode spectrale de Newman et une nouvelle méthode basée sur un algorithme génétique que nous avons développée. Les performances de ces méthodes ont été évaluées en utilisant des réseaux biologiques et artificiels. Leur application n'est toutefois pas limitée aux réseaux de gènes mais peut également se révéler utile pour analyser des réseaux sociaux, technologiques ou encore de neurones. Cependant, les méthodes basées sur l'optimisation d'un critère appelé «modularité» sont connues pour avoir une résolution limitée qui ne leur permet pas de détecter de petites communautés dans de grands réseaux. Plusieurs tentatives ont été proposées pour résoudre ce problème mais aucune d'entre elles n'a permis de résoudre ce problème de manière satisfaisante. C'est pourquoi nous avons développé une méthode permettant de combiner plusieurs structures de communautés pour obtenir une seule structure qui soit à la fois plus robuste et fiable que les structures individuelles. Nous avons montré que cette approche parvient avec succès à surmonter ce problème de résolution. De plus, notre méthode obtient la première place ex aequo avec une autre dans une étude comparant la performance de douze des meilleures algorithmes proposés à ce jour.

La reconstruction de réseaux de gènes impliqués dans le développement d'organismes multicellulaires reste un challenge considérable, puisque ce processus requiert une énorme quantité de données spatiales et temporelles qui n'est généralement pas disponible du fait de la difficulté à mesurer le niveau d'expression des gènes dans un organisme. Une autre contribution de cette thèse est le développement d'une application d'analyse d'images appelée *WingJ* pour la quantification automatique et systématique de l'aile de la drosophile, qui est un modèle classique pour l'étude du développement des tissus. Tout d'abord, un modèle paramétrique de l'aile est construit automatiquement à partir d'une image 3D. La méthode de segmentation est composée de plusieurs modules de détection, chacun ayant pour tâche d'identifier un élément différent de la structure de l'aile ainsi que son orientation. Cette approche modulaire nous a permis d'étendre la méthode de détection à l'embryon de la drosophile. Le modèle de la morphologie ou structure de l'aile identifié est ensuite utilisé pour mesurer et cartographier l'expression des gènes. La représentation obtenue permet de comparer les domaines d'activité des gènes dans des ailes normales et mutantes ou dans des ailes imagées à différents moments de leur développement. Par ailleurs, une représentation robuste et multi-échelles de l'aile de la drosophile est générée en combinant les données morphologiques et d'expression des gènes de plusieurs ailes, complétée par le résultat d'une méthode de détection des noyaux de cellules également développée dans le cadre de cette thèse. Nous avons utilisé la méthode décrite ci-dessus pour générer de manière automatique un modèle robuste d'ailes de drosophiles normales et mutantes imagées à 80, 90, 100, et 110 heures après le dépôt de l'œuf. Finalement, nous avons montré que ces modèles quantitatifs peuvent être utilisés pour reconstruire un réseau d'interactions contenant six gènes responsable en partie du développement de l'aile.

Mots-clés : réseaux de gènes, détection de modules, intégration de données, ingénierie inverse, aile de drosophile, segmentation d'image, modèles multi-échelles, outils informatiques

Contents

1 Introduction

Here, we describe the problem of gene network reconstruction or *reverse engineering*. We then motivate the development and implementation of a comprehensive method for inferring biological organisms from automatic and systematic quantification of morphological and expression information to the generation of *in silico* multiscale models of the organism. We subsequently describe the state of the art of the different methods considered, which are the building blocks of this thesis. We also give a brief introduction to the development of the *Drosophila* wing which is used as a case study. Finally, we state the main contributions of this thesis and give an overview of the following chapters.

1.1 Motivation and challenges

The expression of genes is controlled by regulatory networks, which perform specific functions in a cell. Gene networks play a crucial role in the development of multicellular organisms by precisely coordinating spatial and temporal gene expression patterns during developmental stages. Unravelling and modelling these networks is of key importance to gain eventually a complete understanding of developmental processes and genetically related diseases.

Over the last decade, high-throughput assays for mRNA expression have opened the door to the inference of regulatory networks by allowing simultaneous measurements of the expression levels of thousands of genes. Technologies such as spotted microarrays[1] and oligonucleotide chips[2] have enabled genome-wide quantification of differential gene expression profiles and, more recently, short read sequencing technologies such as RNA-seq[3] have provided more precise quantification of mRNA levels. Since then, a plethora of methods have been proposed to reverse reverse genetic networks in single cells[4]. However, accurate and systematic evaluation of these methods is hampered by the difficulty of constructing adequate benchmarks and the lack of tools for a differentiated analysis of network predictions on such benchmarks.

The reconstruction of a developmental gene network in its spatial context, by opposition to single-cell gene networks, remains a considerable challenge. One of the reason is that this process requires tremendous amount of spatial and temporal gene expression data, which are usually available in very limited quantities due to the inherent difficulty in measuring gene expression in an entire organism.

These challenges bring about the main focus of this thesis:

- **Evaluation of network inference methods.** The challenge addressed is the development of methods and computational tools for the generation of biologically plausible *in silico* networks to enable the performance profiling of inference methods on such benchmarks.

- **Unsupervised and systematic quantification of biological organisms.** This challenge consists in the development of an automatic detection and segmentation method for generating quantitative description of biological systems (organ or body systems). These descriptions should provide reliable information about the morphology of the system as well as the expression levels of genes of interest in order to provide meaningful targets to the reverse engineering algorithm.

Another challenge addressed by this thesis is the development and implementation of extensible and user-friendly computational tools. In particular, the objective is to provide biologists and computer scientists with tools they can both use to gain insight into their own research topic.

1.2 State of the art

1.2.1 Benchmark generation for network inference methods

Numerous methods have been developed for inference of gene regulatory networks, however relatively little effort has been put into evaluating the performance of those methods on adequate benchmarks. So far, three main strategies have been proposed to generate benchmark networks. A first strategy consists in evaluating network predictions made by reverse engineering algorithms on well studied *in vivo* pathways from model organisms[5,6]. However, those networks are incomplete maps of the physical interactions in the cell that are responsible for cellular functions and using them as benchmarks imply making error when evaluating network predictions. Another strategy consists of genetically engineering synthetic *in vivo* networks[7,8]. The main drawback of this strategy is that only a few small networks are available. Yet another strategy consists in developing *in silico* gene regulatory networks that can be simulated to produce artificial gene expression data. The simulation of *in silico* networks has the advantages of being fast, easily reproducible, and less expensive than biological experiments. A few instances of small *in silico* networks with handcrafted topologies[9] have been proposed as benchmarks for reverse engineering algorithms. More recently, several generators have been developed to automate the construction of *in silico* regulatory networks including up to thousands of genes to be used as benchmark networks for reverse engineering algorithms[10–12].

Benchmark generators such as AGN[10] aim to produce *in silico* gene networks exhibiting topological properties observed in biological networks using Erdös-Renyi, Watts-Strogatz (small-world), or Albert-Barabási (scale-free) random graph models. However the structures generated using random graphs capture only few of the structural properties of gene regulatory networks[11] and do generally not display important properties such as modularity[13] or occurrences of network motifs, which are statistically over-represented regulatory patterns in biological networks[14]. Instead of constructing more complex random structures based on graph theory, which may be difficult to justify[10], SynTReN[11] and ReTRN[15] chose to generate network structures by extracting parts of known *in vivo* regulatory network structures. This approach has the advantage of capturing several structural properties observed in *in vivo* network structures[11].

In order to produce gene expression data, the generated structures must be endowed with dynamical models of gene regulation. Systems of non-linear ordinary differential equations (ODE) are widely used[16,17], but other approaches exist[12]. ODE systems allow to continuously describe levels of gene products and rates of reactions taking place in the network models where biological processes that have not been fully characterized yet are abstracted. Because current high-throughput technologies do not allow the monitoring of protein expression as microarrays do for RNA[12], some benchmark generators consider mRNA as a proxy for protein expression and thus do not model translation independently of transcription[11,15]. Protein expression, however, does not correlate perfectly with mRNA expression in real biological systems due in part to different degradation rates of mRNA and protein products[18]. RENCO[16],

GeNGe[17], and GRENDEL[19] are examples of available benchmark generators considering both transcription and translation processes in their respective dynamical models.

1.2.2 Community structure detection in complex networks

Biological interaction networks are often organized into groups or modules of related genes and proteins carrying out specific biological functions. Over the past few years, many methods have been proposed to rationally decompose these networks into meaningful or functional modules, yet this challenging problem has not yet been successfully solved.

Many systems can be described as complex networks where nodes represent individual units connected by edges depending on their relative interaction or relationship. The structure of a complex network has features that do not occur in random networks but often occur in real graphs. Examples include metabolic, protein and genetic interaction networks as well as neural, social and technological networks[20–23]. In such networks, there are usually groups of nodes that are more highly connected to each other than to the rest of the network. Such groups are usually called *modules*, *communities* or *clusters*[20,24]. In social networks, nodes often represent people connected by friendship relations and detecting communities (groups of friends or people with the same interest, family, etc.) have numerous applications[20]. In protein-protein interaction (PPI) networks, there are often groups of proteins that interact tightly in a same complex[6,21]. Therefore, the identification of these modules is of great importance to unravel the structural and functional properties of these networks.

The inspiration for developing novel community structure detection methods has been taken from diverse fields such as physics, biology, applied mathematics, and computer sciences, for instance[25]. Most of the existing algorithms are based on the maximization of a quantity called *modularity*, which is used to estimate the goodness of a partition of the network into modules based on the comparison between the partitioned network and a randomized version of this network that has the same node degree distribution[26,27]. Different techniques have been proposed to find partitions that optimize the modularity. Spectral methods usually perform an eigendecomposition of the adjacency matrix (the matrix that describe the structure of graph) or Laplacian matrix in order to describe the solution of the problem as a linear combination of their eigenvectors[27–29]. Hierarchical algorithms target the decomposition of the network into a hierarchy of communities[13,30] based either on an agglomerative (each node initially belongs to its own community) or divisive (all nodes initially belong to the same community) approach. Another technique consists in applying greedy or stochastic optimization algorithms to find the partition of the network that obtain the maximum modularity value[31–34].

So far, one of the best performer method called *Infomap* has been developed by Rosvall & Bergstrom. The idea behind this method is to compress optimally the information obtains from multiples random walks taking place in the network. The size of the modules inferred are then proportional to the average time a random walker spends on nodes in the module[35].

This is carried out rather quickly using a deterministic greed search algorithm[31]. The result is then refined using a simulated annealing approach[32].

1.2.3 Quantification of multicellular organisms for gene network inference

In this thesis, we are interested in the automatic detection and segmentation of multicellular systems, namely the *Drosophila* wing pouch (**Section 1.3**). The process consists in detecting the system of interest in a stack of confocal fluorescence images before generating a model that describes morphological and gene expression information. If the detection, tracking and gene expression quantification of single-cell organisms has been largely addressed[36–38], the problem of generating a quantitative description of multicellular system such as an entire organism or organ for gene network reverse engineering is relatively recent.

The only work that shares some similarity with the contribution presented in **Chapter 4** has been recently published by Crombach & Wotton[39]. The method they propose consists first in detecting the *Drosophila* embryo at different time points during development. The detection is performed using gamma adjustment, Sobel edge detection, and filling of holes. The remaining *blobs* are then used as masks to extract the embryo from the images. Embryos are then manually rotated to place them in a canonical orientation. Embryos are then skeletonized to identify a line that runs along the anterior-posterior axis of the embryo, which is then used to measure mRNA expression data. A reverse engineering algorithm previously proposed by Jaeger[40] is finally applied to reconstruct the gap gene network[41].

1.3 Introduction to the development of the *Drosophila* wing

The adult appendages of *Drosophila* such the wings, legs, antennae and halteres (the balancing organs of the fly) are derived from *imaginal discs*[42]. These imaginal discs form in the embryo as a small cluster of cells[43]. During the growth phase, the imaginal discs are mainly composed of a single-layered sheet of columnar cells, which is contiguous to another layer of cells called the peripodial membrane (**Fig. 1.1**). There are two discrete stages that metamorphose the discs during larval development. During growth, patterning of the discs is dependent on secreted molecules called morphogens. Examples of morphogens include Decapentaplegic (Dpp) and Wingless (Wg) which are both required for proper development of the wing imaginal discs[44,45]. The number of cells increase by approximately a 1000 fold in four days (during the larval period) to reach nearly 50'000 cells at a stage of development called *late third instar*[43].

The wing disc includes a region called the *wing pouch*. In the adult wing, the wing pouch gives rise to the wing blade while the part surrounding it (called hinge) forms a flexible link attaching the wing blade to the body wall of the fly. The wing pouch is divided by the anterior/posterior (A/P) and dorsal/ventral (D/V) compartment boundaries into four compartments[46]. The relation between these four compartments of the pouch and the adult wing is introduced in

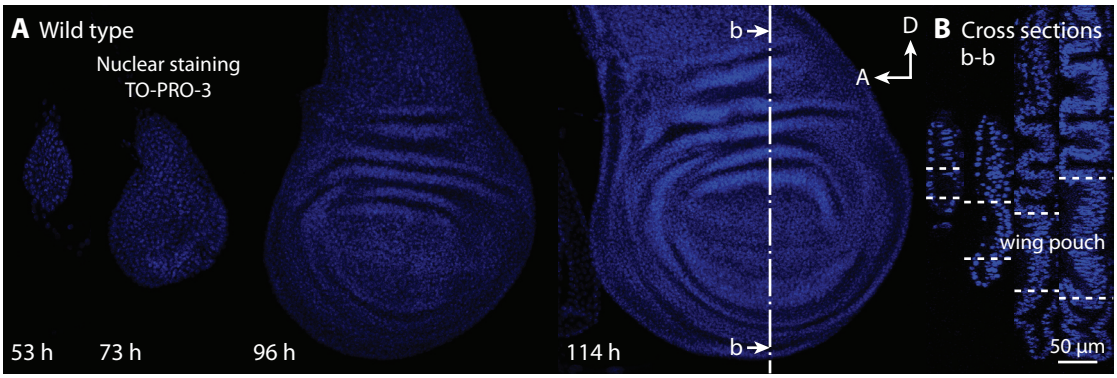


Figure 1.1: Visualization of cell nuclei in *Drosophila* wing disc with TO-PRO marker. **(A)** In our experiments, the nuclei of the columnar cells that compose the wing imaginal disc were stained with TO-PRO-3 for time points between 53 and 114 hours after egg laying (AEL). **(B)** The cross section of the wing discs in panel **(A)** shows that the imaginal disc is single-layered and flat in the region of the pouch for the time points considered **Video S4**, and that it is composed of columnar cells whose nuclei are at different z locations.

Video S4. We use the expression of Wingless (Wg) labelled with antibodies to visualize the contour of the wing pouch and the D/V boundary (blue channel in **Fig. 1.2**). In a similar way, the A/P boundary can be identified via the expression of Patched (Ptc, also visible on the blue channel). During eversion, the single-cell layered wing pouch everts to give rise to the *double-layered adult wing*[47]. We consider that the *Drosophila* wing is a model uniquely suited for a systems biology approach and for studying the genetic program that governs the growth and shape of an organ[45,48,49].

1.4 Original contribution

Methods

The contribution of this thesis is two-fold. The first contribution consists in the development and evaluation of many methods in various research fields including computational biology, artificial intelligence, optimization algorithms, graph and network theory, and image analysis.

First, we present a novel and comprehensive method for *in silico* benchmark generation and performance profiling of network inference methods (**Fig. 1.3A**). The structure of inferred gene networks can then be rationally decomposed into function modules using an extensible and modular framework for community structure detection in complex networks with applications to neural, social, and technological networks (**Fig. 1.3B**). Moreover, we present a framework for unsupervised detection and segmentation of the morphology of biological organisms and organs, quantification of gene and protein expression, and detection of cell nuclei (**Fig. 1.3C**). Heterogeneous datasets are then combined to produce robust and reliable

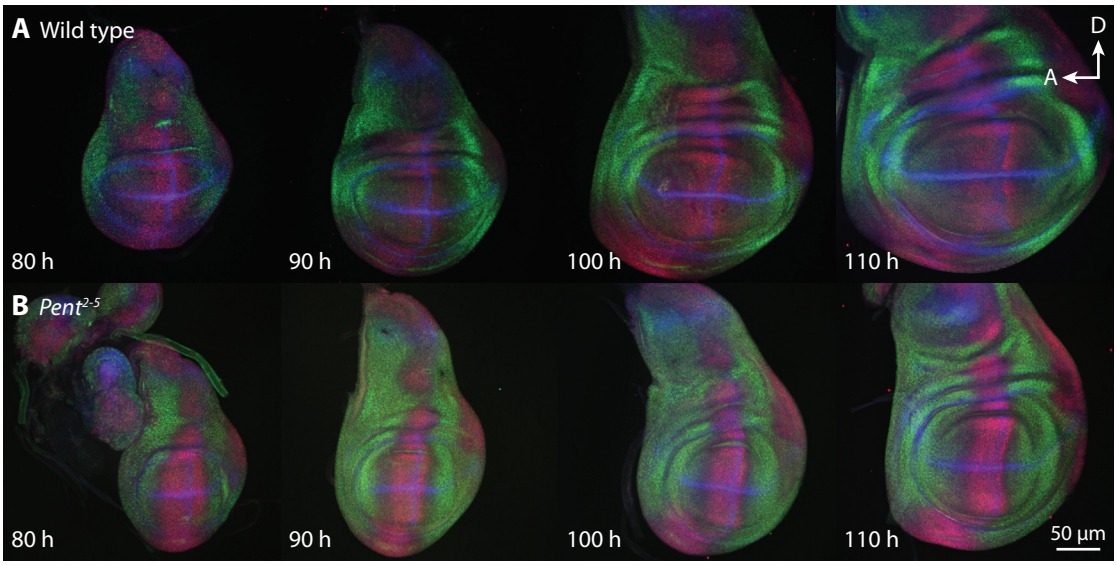


Figure 1.2: Wild type and *pent*$^{2-5}$ *Drosophila* wing imaginal discs. (A) Wild type and **(B)** *pent* deficient discs imaged at 80 and 110 hours AEL (late third instar). The wing pouch structure is defined by the expression of Wg-Ptc-AB (blue). The expression of Pmad-AB (red) and Brk-AB (green) are also reported. The images illustrate the effect of the mutation *pent*$^{2-5}$ on the Dpp gradient activity, which has been shown to play a role in the growth and patterning of the *Drosophila* wing[50]. **Video S4** illustrates the development of the *Drosophila* wing with 3D rendering of confocal fluorescence images and a 3D animation.

quantitative descriptions of the biological system to reverse engineer. Specifically, the main research contributions of this thesis are

- a method for generating biologically plausible *in silico* networks for performance profiling of network inference methods (**Chapter 2**);
- performance profiling of state-of-the-art inference method using standard metrics and the output of a network motif analysis;
- identification of the most informative type of gene expression data to provide to a given inference method to achieve the best possible reconstruction from *in vivo* experiments;
- validation of hundreds of gene network reconstructions generated by 91 teams in the context of three DREAM challenges (DREAM3, DREAM4, and DREAM5). Today, the accuracy of more than 25,000 gene network reconstructions have been evaluated by GNW users;
- an extensible and modular method for identifying the community structure of complex networks (**Chapter 3**);

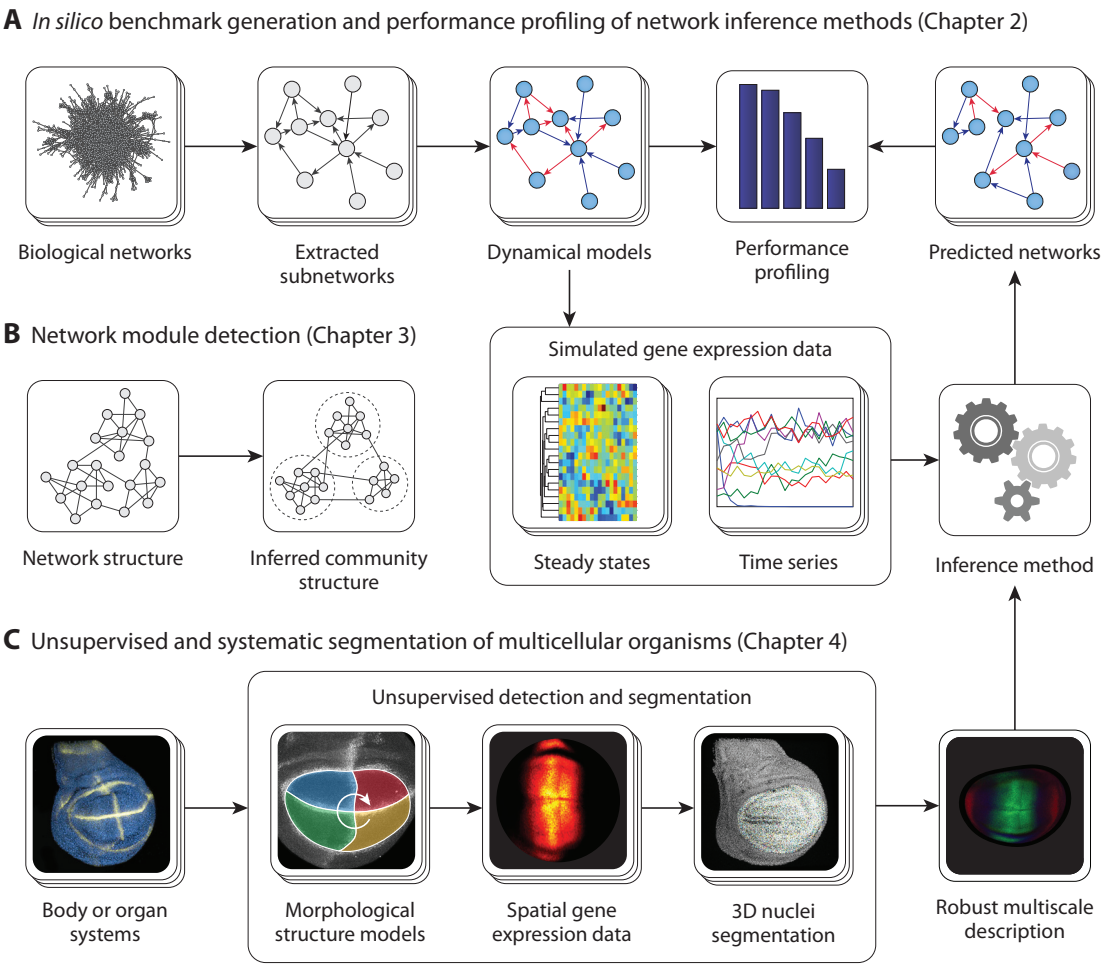


Figure 1.3: Comprehensive framework for reverse engineering models of biological organisms. (A) *In silico* benchmark generation and performance profiling of gene network inference methods. **(B)** After reconstructing the network of interactions between genes and transcription factors from expression data, an additional level of insight is gained by rationally decomposition the network into function modules. **(C)** Automatic detection and segmentation algorithms for generating robust and multiscale quantitative descriptions of biological organisms. These descriptions then provide meaningful targets for reverse engineering models that account for the development of multicellular organisms.

- a GA-based modularity optimization method to unravel the community structure of complex networks;
- validation of several community structure detection methods on real and artificial networks;
- a community voting method that enables to overcome to some extend the *resolution limit* of modularity optimization methods;

- initiation and supervision of an interdisciplinary project between the Laboratory of Intelligent Systems and Biomedical Imagining Group at EPFL, and the Affolter Lab at University of Basel (**Chapter 4**);
- an unsupervised detection and segmentation method to generate parametric models that describe the morphology or structure of the *Drosophila* wing pouch and embryo from stacks of confocal fluorescence images;
- a procedure to measure mRNA and protein concentration levels in a systematic way in the space. Collected datasets are then used to build a novel representation called *gene expression maps*;
- an unsupervised 3D cell nuclei detection and segmentation methods based on a *watershed algorithm*;
- a method to integrate morphological and expression datasets into a single, robust and reliable quantitative description of biological organisms and organs;
- generation of quantitative descriptions of the *Drosophila* wing pouch at different time points during development;
- quantification of the effect of the $pent^{2\text{-}5}$ mutation, which inhibits the growth of the *Drosophila* wing, on the wing morphology and domains of activation of several genes.
- reverse engineering of six-gene network that participate to the growth and patterning of the *Drosophila* wing using expression datasets collected using the above method;

Software applications

Another important contribution of this thesis is the development, implementation, and immediate availability of several software applications. All the methods that we developed and introduce in the following chapters can already be used to support the research of other groups. To achieve this, the following software applications have been designed with always keeping in mind the objective of making them extensible, user-friendly, well documented, and useful to a broad community of biologists and computer scientists (**Fig. 1.4**).

The computational tools developed in the context of this thesis are

- a software application called *GeneNetWeaver (GNW)* that provides for the first time tools for *in silico* benchmark generation and performance profiling of network inference methods. See tschaffter.ch/projects/gnw;
- a software application called *Jmod* that implements several state-of-the-art community structure detection methods. In addition to provide a framework for the development of additional methods, Jmod provides tools to gain insight into the behavior of community

Figure 1.4: Extensible and user-friendly computational tools for quantifying and reverse engineering biological systems. These tools are provided with an extensive documentation that includes user manuals, video tutorials, APIs (application programming interfaces), and other supporting data. Source code is also widely documented and available under open source license, which allows other researchers and engineers to further extend our work.

structure detection methods, thereby indicating potential ways of improving inference methods. See tschaffter.ch/projects/jmod;

- a software application called *WingJ* for unsupervised detection and segmentation of the *Drosophila* wing pouch and embryo from stacks of confocal images, including cell nuclei detection and integration of datasets collected from multiple wings, for instance. See tschaffter.ch/projects/wingj;

- a software application to track the genetic identify of heterogeneous flies walking in a transparent chamber. This project aims to develop quantitative behavioral methods to understand brain function and evolution in fruit fly (*Drosophila melanogaster*). This work is currently conducted by Dr. Pavan Ramdya. See tschaffter.ch/projects/squid;

- Java library that provides several algorithms to numerically integrate stochastic differential equations, which we used to simulate molecular noise in the dynamics of *in silico* transcriptional gene regulatory networks. See tschaffter.ch/projects/libsde.

Table 1.4 provides a few metrics about the software applications introduced in this thesis. The number of lines of code gives a relative good idea of the effort spent in the different research projects considered in this thesis. The source code is also particularly well documented as indicated by the number of comments per line of code, which makes it easier for researchers and engineers to further extend our work.

Table 1.1: Source metrics of the computational tools developed during the thesis. The metrics are the number of physical lines of code (LOC, does not include white and comment lines), the number of comments, the ratio [Num. comments]/[LOC], the number of classes, and the number of methods. The metrics have been generated using *Google CodePro AnalytiX*.

Software application	LOC	Comments	Comment ratio	Classes	Methods
WingJ	21440	6400	29.8	131	1352
GNW	20204	4573	22.6	124	1603
Jmod	9066	2715	29.9	81	716
sQuid	6086	2130	35	34	548
libSDE	669	301	44.9	7	73

Author contributions

The realization of this thesis benefited from and led to some fruitful collaborations. Regarding the content of the following chapters, I have developed the methods, designed and performed the experiments, and designed and implemented the software applications, with the following exceptions. The comprehensive method for *in silico* benchmark generation and performance profiling of network inference methods has been co-developed with Daniel Marbach. Daniel designed the activation function $f(\cdot)$ used in the dynamics of *in silico* networks and proposed the analysis of network motifs. I have implemented GeneNetWeaver with a significant contribution from Daniel. Gilles Roulet and Jonathan Rohrbach also contributed through two outstanding student projects to the network evaluation part and the implementation of a multicellular extension for GNW, respectively. I have initiated the collaboration between the Laboratory of Intelligent Systems and the Biomedical Imaging Group at EPFL, and the Affolter Lab at University of Basel. Ricard Delgado-Gonzalo provided invaluable expertise for the development of active contour algorithms (also called *snakes*) and their formal definition. He also significantly contributed to the implementation of WingJ. Finally, Fisun Hamaratoglu provided us with stacks of confocal fluorescence images of the *Drosophila* wing.

1.5 Organization of the thesis

The dissertation is organized in six chapters, including the introduction (**Chapter 1**) and the conclusion (**Chapter 5**). At the beginning of each chapter, an abstract is given to situate the chapter in the context of the thesis and summarize its content.

- **Chapter 2** ***In silico benchmark generation and performance profiling of network inference methods.*** This chapter introduces a novel and comprehensive method for generating biologically plausible *in silico* networks and evaluating the performance of network inference methods available to the community as an open source software called *GeneNetWeaver (GNW)*. Using GNW, we evaluate the performance of six inference methods and perform a network motif analysis to identify systematic prediction errors.

We also show how the performance of those inference methods are affected by the structural properties and the size of the gene regulatory networks to infer, and how GNW can help to identify the most informative type of gene expression data to provide to a given inference method to achieve the best possible reconstruction from *in vivo* experiments. Moreover, we have used GNW to evaluate the accuracy of more than 900 network predictions generated by 91 teams that have participated to the international DREAM3, DREAM4, and DREAM5 challenges.

- **Chapter 3** ***Extensible and modular community structure detection in networks.*** In this chapter, we present an extensible and modular method to infer the community structure of biological networks. This method is implemented as an open source Java toolkit called *Jmod*. We introduce an improved version of Newman's algorithm, a GA-based modularity optimization method, a brute force approach, and two refinement techniques called MVM and gMVM. The performance of the methods is evaluated on real and artificial networks. Moreover, we show that the performance of modularity optimization algorithms is affected by a *resolution limit* that makes them fail to identify small communities in large networks. To overcome this limitation, we have developed a community voting to combine multiple community structure predictions into a more reliable and robust prediction. Using the GA-based method, we show that our approach is best performer in a comparative analysis that profiled the performance of twelve state-of-the-art community structure detection algorithms.

- **Chapter 4** ***Towards unsupervised and systematic segmentation of biological systems.*** We have developed an unsupervised and systematic method to generate quantitative description of biological organisms, which we applied to the developing *Drosophila* wing. First, we present a fully automated detection and segmentation method that infer a parametric model of the morphology or structure of the wing pouch from stacks of confocal fluorescence images. This model then provides a coordinate system to measure mRNA and protein concentration levels in a systematic way in 2D (possibly 3D). The heterogeneous datasets collected for each wing are then integrated using an automatic procedure to obtain a robust and multiscale quantitative description of the wing. This description is further extended with the output of an unsupervised cell nuclei detection and segmentation method that we developed. The above method is available as an extensible, user-friendly and open source software called *WingJ*. Using WingJ, we have quantified the wing pouch at different time points during development. We also produce experimental data that show how the $pent^{2\text{-}5}$ mutation inhibits the growth of the wings. Finally, we use the datasets collected using WingJ to reverse engineer a six-gene regulatory network that participates to the developed of the *Drosophila* wing.

- **Appendices.** The first appendix provides the links to the different project websites. It also lists and summarizes its content of several supplementary videos edited to illustrate this thesis. Then, three appendices contain supplementary information for **Chapter 2**, **Chapter 3**, and **Chapter 4**. The following appendix gives a brief introduction to stochastic differential equations (SDEs) which we have used to model molecular noise in the

dynamics of *in silico* gene networks. The two last appendices include the list of the publications written during the thesis and my resume.

2 *In silico* benchmark generation and performance profiling of network inference methods

Over the last decade, numerous methods have been developed for inference of regulatory networks from gene expression data. However, accurate and systematic evaluation of these methods is hampered by the difficulty of constructing adequate benchmarks and the lack of tools for a differentiated analysis of network predictions on such benchmarks.

Here we describe a novel and comprehensive method for *in silico* benchmark generation and performance profiling of network inference methods available to the community as an open-source software called *GeneNetWeaver (GNW)*[a]. In addition to the generation of detailed dynamical models of gene regulatory networks to be used as benchmarks, GNW provides a network motif analysis that reveals systematic prediction errors, thereby indicating potential ways of improving inference methods. The accuracy of network inference methods is evaluated using standard metrics such as precision-recall and receiver operating characteristic (ROC) curves. We show how GNW can be used to assess the performance and identify the strengths and weaknesses of six inference methods. Furthermore, we used GNW to provide the international DREAM (Dialogue for Reverse Engineering Assessments and Methods) competition with three network inference challenges (DREAM3, DREAM4, and DREAM5).

This chapter is based on the publication *GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods*[51].

[a] tschaffter.ch/projects/gnw

2.1 Introduction

A challenging issue in systems biology is the development of computational tools for the reverse engineering of gene regulatory networks from quantitative experimental data. Over the last decade, high-throughput assays for mRNA expression have opened the door to the inference of regulatory networks by allowing simultaneous measurements of the expression levels of thousands of genes. Technologies such as spotted microarrays[1] and oligonucleotide chips[2] have enabled genome-wide quantification of differential gene expression profiles and, more recently, short read sequencing technologies such as RNA-seq[3] have provided more precise quantification of mRNA levels.

Here we describe a method for *in silico* benchmark generation and performance profiling of network inference methods available to the community as an open-source software called GeneNetWeaver (**Fig. 2.1**). GNW has an intuitive graphical user interface that makes the generation and simulation of gene network models as simple as a few clicks. Network topologies are generated by extracting modules from known *in vivo* gene regulatory network structures such as those of *E. coli*[6] and *S. cerevisiae*[52]. These structures are then endowed with detailed dynamical models of gene regulation including both transcription and translation processes using a thermodynamic approach accounting for both independent and synergistic interactions[53]. Expression data can be generated either deterministically or stochastically to model molecular noise in the dynamics of the networks, and experimental noise can be added using a model of noise observed in microarrays[54]. Different types of *in vivo* experimental procedures, such as wild type, knockout (null-mutant), knockdown (heterozygous), and multifactorial perturbations, can be reproduced by the software. In addition, a unique feature of GNW is the systematic and comparative evaluation of predictions by different inference methods, which none of the existing benchmark generators provide. GNW performs an exhaustive network motif analysis for a set of network predictions, which often reveals systematic prediction errors, thereby indicating potential ways of network reconstruction improvements. The accuracy of network inference is also assessed using standard metrics such as precision-recall and receiver operating characteristic (ROC) curves.

Furthermore, we show how GNW can be used to generate *in silico* benchmark suites to assess the performance and identify strengths and weaknesses of six network inference methods. We also show how the performance of those inference methods are affected by the structural properties and the size of the gene regulatory networks to infer, and how GNW can help to identify the most informative type of gene expression data to provide to a given inference method. Finally, we assess the performance of those six inference methods on the network inference challenge that we provided to the international DREAM4 competition (Dialogue for Reverse Engineering Assessments and Methods).[b]

[b] the-dream-project.org

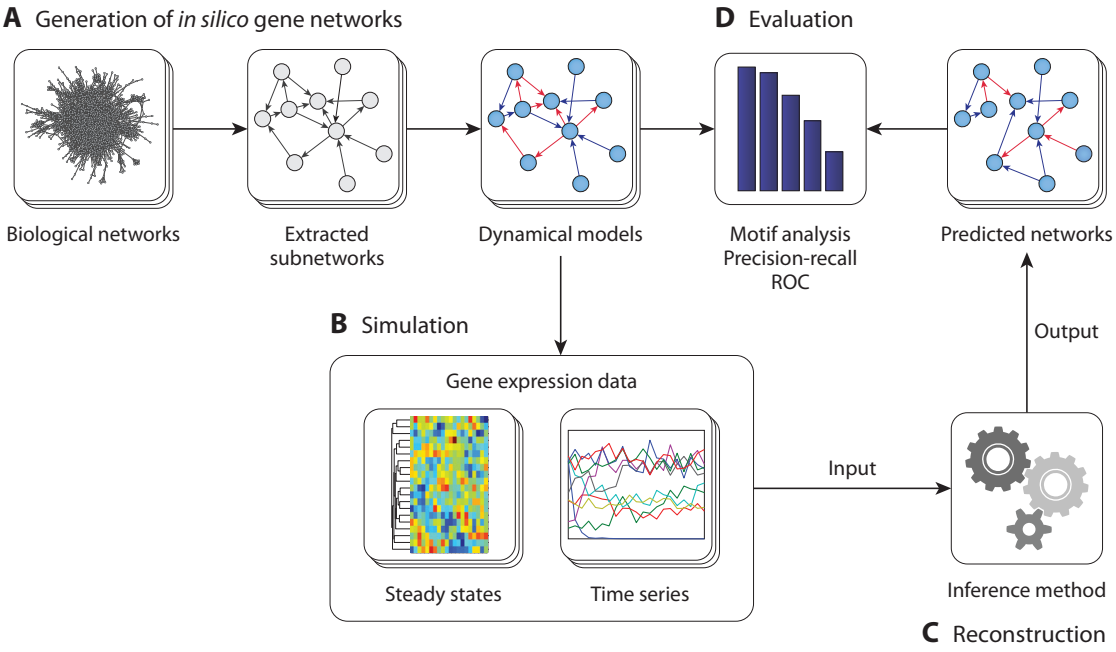


Figure 2.1: Benchmarking and performance assessment of network inference methods using GNW. (A) *In silico* gene networks are obtained by extracting subnetwork structures from known transcriptional networks (*E. coli*, *S. cerevisiae*, etc.) before being endowed with detailed dynamical models of gene regulation accounting for both transcription and translation, independent and synergistic interactions, as well as molecular and measurement noise. **(B)** *in silico* gene networks are simulated to produce steady-state and time-series expression data for a variety of experiments such as wild-type, knockout, knockdown, and multifactorial perturbation experiments. **(C)** Inference methods are asked to predict structures of *in silico* benchmark networks from gene expression data. **(D)** From network prediction files, GNW performs a network motif analysis which often reveals systematic prediction errors, thereby indicating potential ways of network reconstruction improvements. It also automatically generates comprehensive reports including standard metrics such as precision-recall and receiver operating characteristic (ROC) curves.

2.2 Generation of *in silico* gene networks

2.2.1 Module extraction from global interaction networks

Instead of using random graph models, which are known to only partly capture the structural properties of biological networks[11], we generate network structures by extracting modules from known biological interaction networks such as those of *E. coli*[6] and *S. cerevisiae*[52] (the *source networks*). Our approach is based on the extraction of modules, that is, groups of genes that are more highly connected than expected in a random network[55]. We have shown that the topological modules extracted using our method correlate with functional modules of the source networks[55]. Hence, obtained network structures are meaningful targets for reverse

engineering algorithms because in practice, one typically tries to infer the structure of a set of functionally related genes.

In the next two sections, we describe two methods for extracting subnetworks that capture the structural properties of the source network. In **Section 2.2.1**, the first method called *bottom-up* extracts the subnetwork node by node until the desired size is reached[55]. The second method uses a *top-down* approach to partition the source network into *functional* modules (**Section 2.2.1**). The desired module is then selected based on its size, for instance, to provide the structure of the *in silico* gene network.

Bottom-up module extraction

The bottom-up method has been developed to generate network structures of any size M where $M \leq N$ and N is the size of the source network[55]. This feature is important for *in silico* benchmark generation because network inference methods can not all reconstruct large networks. For example, algorithms based on correlation[56] or mutual information (MI)[57,58] can be applied to reconstruct networks including thousands of genes, however ODE-based methods are limited to the reconstruction of much smaller networks, typically less than a hundred nodes[59,60].

We use a hierarchical, modular and scale-free source network to illustrate the bottom-up extraction of a M-node subnetwork (**Fig. 2.2A**). The module extraction starts from a seed node which is selected either manually or randomly. In **Figure 2.2B**, the seed node is set to the central node of the network, which is also the first node of the subnetwork. Initially, we only consider the neighbors of the seed node as candidates to be included in the subnetwork. Instead of selecting it randomly as shown in **Figure 2.2C**, we use the first part of Newman's spectral algorithm[27] which consists in finding the partition of a group of nodes into two communities that maximizes a quantity called modularity Q. The modularity Q is defined as

Q =(fraction of edges falling within modules)
- (expected fraction of such edges in randomized graphs)

and takes usually values in $[-0.5, 1)$ where values near 1 means that the network is highly modular (see **Section 3.2.1**). Here our iterative algorithm always considers two communities: the subnetwork which is being extracted and the remaining group of nodes of the source network. Lets assume that the seed node has n' neighbor nodes. We successively and independently add one of them to the subnetwork before computing the Q values of the n' splits of the source network into the 2-node subnetwork and the group that contains the $n-2$ remaining nodes. The first iteration of our algorithm ends after the neighbor node associated to the largest Q value is effectively added to the subnetwork.

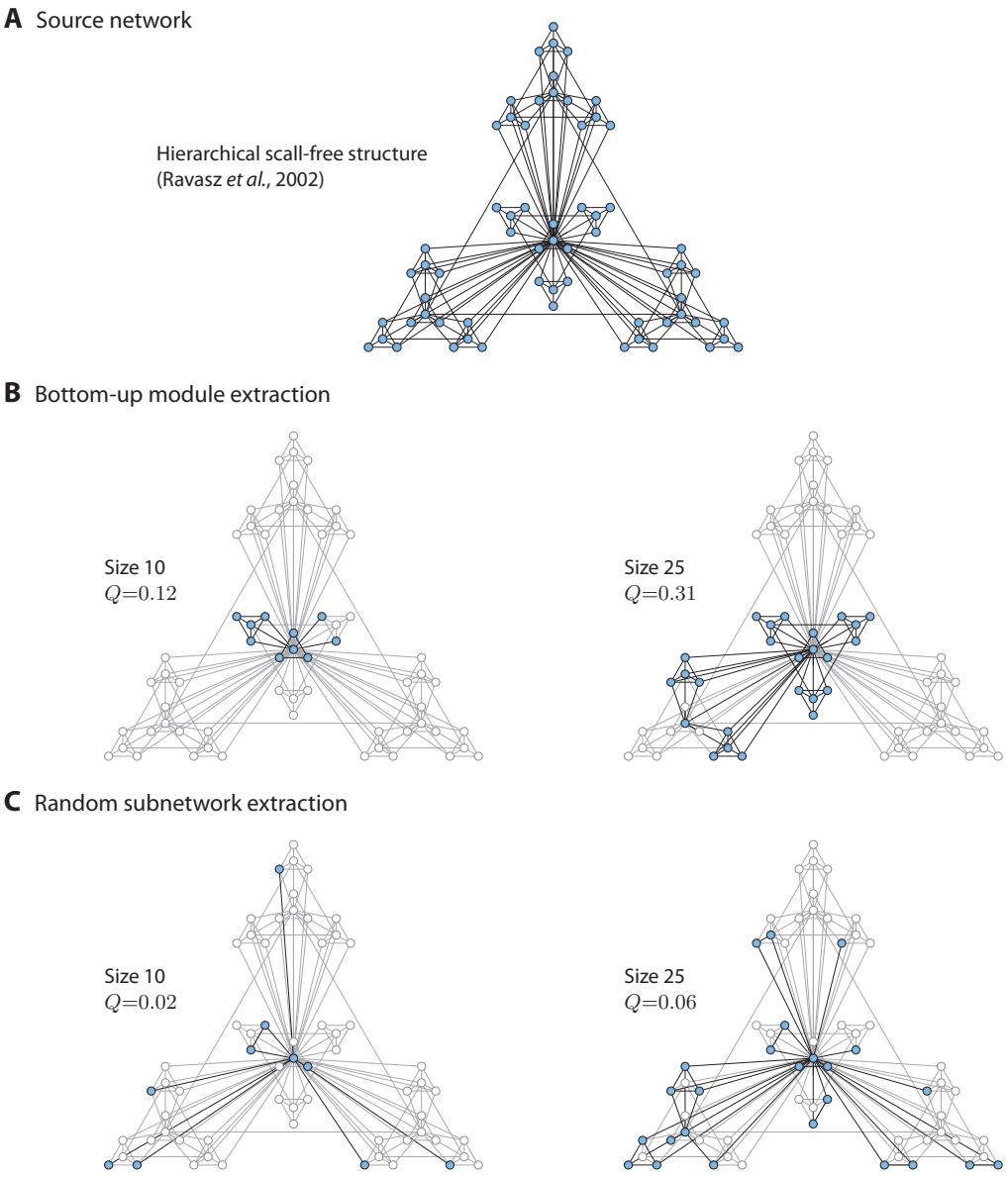


Figure 2.2: Bottom-up module extraction from interaction networks. **(A)** We illustrate the extraction using a hierarchical, modular and scale-free source network[13]. This network is available in GNW and can be used to visualize the effects of the parameters of the module extraction method. **(B)** The extraction starts from a *seed node* either selected manually or randomly. We use the first part of Newman's spectral algorithm[27] to ensure that the node added to the subnetwork increase the modularity value Q associated to the organization of the source network in two communities: the subnetwork and the group that contains the remaining nodes of the network. At each iteration, all the neighbor nodes that share at least one connection with the subnetwork are listed before we select the one that maximize Q. **(C)** Network structures generated by randomly picking nodes clearly can not capture the structural properties of the source network.

For the second iteration, all the neighbor nodes that share at least one edge with the 2-node subnetwork are selected before selecting the one that maximize Q as performed previously. The algorithm eventually ends when the specified size of the subnetwork has been reached.

We observe in **Figure 2.2B** that the subnetwork extracted conserves the natural modules of the source network. We also have shown that the number of different triads (three-node motifs which can be seen as the building blocks of a network) in the extracted subnetworks are proportional to the total number of triads found in the source network[55], which is not the case in randomly extract subnetworks (**Fig. 2.2C**).

It is important to distinguish between the modularity of the subnetwork and the Q value computed to split the source network in two communities. Assuming that the seed node falls initially in c-node module or community of the source network, the subnetwork should have a high modularity value if its size is smaller or equal to c because its structure is one part of a densely connected group of nodes. However, the structure of the subnetwork would still reflect the 3-node motifs organization of the source network. Furthermore, we provides parameters to add some randomness in the extraction process. Instead of selecting systematically the node that maximize the Q value, we can specify that the algorithm should pick randomly one of the best 10% nodes, for instance. Another advantage is that this can make the generation of different network structures much easier depending on the size and structural properties of the selected source network.

Top-down module extraction

In addition to the above bottom-up algorithm, we propose a second approach for extracting subnetworks that capture the structural properties of the source network (e.g. known biological networks). The top-down strategy consists in first identifying the modules or communities that compose the source network before selecting one of them to provide the structure of the *in silico* gene network. The extracted structures are particularly meaningful targets for reverse engineering algorithms because in practice, one typically aims to infer the structure of a set of functionally related genes. **Figure 2.3** shows the community structure of the *E. coli* transcriptional network[6]. This network corresponds to the TF-gene of RegulonDB release 6.7[c]. Therefore, each node represent a transcription factor or gene and the edge the interaction between two elements.

Chapter 3 is actually dedicated to the detection of the community structure in complex networks, that is, the inference of the modules or communities that compose the network. Here we applied a GA-based detection method (**Section 3.2.2**) to identify the functional modules of the *E. coli* network. From the list of identified modules, one of them is picked based on its size to provide the structure of the *in silico* gene network. Actually, the bottom-up method introduced previously can be used to increase but also decrease the size of the

[c] regulondb.ccg.unam.mx

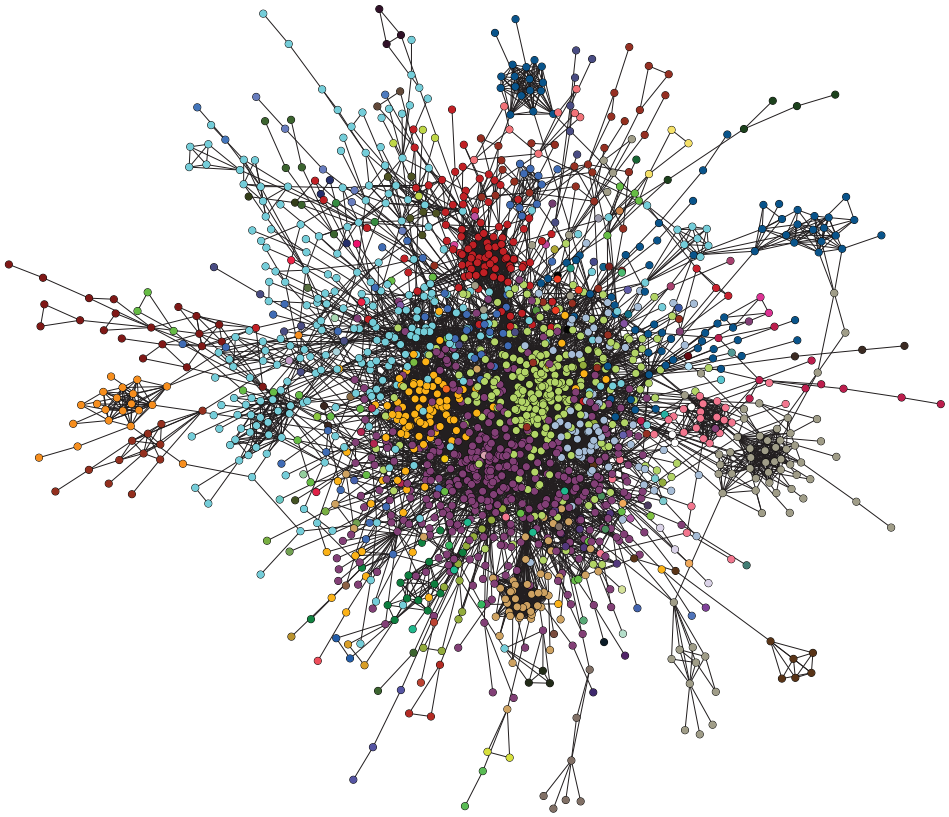


Figure 2.3: Top-down module extraction from the *E. coli* transcriptional network. This network includes 1565 transcription factors (TFs) and genes interacting through 3758 connections (RegulonDB release 6.7). We apply a novel GA-based community structure detection method to identify the functional modules of *E. coli*. One of them is then selected to provide the structure of the *in silico* network. Smaller or larger structures can be obtained by adding or removing nodes one by one using the bottom-up extraction method (**Section 2.2.1**). Much larger structures can also be generated by merging several neighbor modules. Nodes belonging to different modules are painted in different colors. The visualization of the network is obtained using a force-directed layout[61].

structure in order to achieve the specified size. Moreover, larger structures can be produced by merging several neighbor modules.

2.2.2 Modeling the dynamics of transcriptional gene regulatory networks

Network topologies are endowed with detailed dynamical models of gene regulation. Both transcription and translation are modeled using a standard thermodynamic approach[53] allowing for both independent ("additive") and synergistic ("multiplicative") regulatory interactions. For each gene i of a network, the rate of change of mRNA concentration F_i^{RNA} and the rate of change of protein concentration F_i^{Prot} are described by

$$F_i^{RNA}(\boldsymbol{x},\boldsymbol{y}) = \frac{dx_i}{dt} = m_i \cdot f_i(\boldsymbol{y}) - \lambda_i^{RNA} \cdot x_i \tag{2.1}$$

$$F_i^{Prot}(\boldsymbol{x},\boldsymbol{y}) = \frac{dy_i}{dt} = r_i \cdot x_i - \lambda_i^{Prot} \cdot y_i \tag{2.2}$$

where m_i is the maximum transcription rate, r_i the translation rate, λ_i^{RNA} and λ_i^{Prot} are the mRNA and protein degradation rates, and $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors containing all mRNA and protein concentration levels, respectively. $f_i(\cdot)$ is the activation function of gene i, which computes the *relative activation* of the gene, which is between 0 (the gene is shut off) and 1 (the gene is maximally activated), given the protein or transcription-factor (TF) concentrations $\boldsymbol{y}$. A more detailed description of the activation function used is given by[62]. Note that our approach conserves the nature of the gene interactions (enhancing or inhibitory) of the imported or extracted network structures.

The integration of the system of equations defined by (2.1) and (2.2) results in noiseless mRNA and protein concentration levels, respectively $x_i(t)$ and $y_i(t)$ for gene i (see **Appendix B.1.3**). In living cells, molecular noise originates from thermal fluctuations and noisy processes such as transcription and translation[63]. Hence, random fluctuations affect concentration levels of mRNA and protein, whose expression can be viewed as a stochastic process[64]. Both F_i^{RNA} and F_i^{Prot} are of the form

$$\frac{dX_t}{dt} = V(X_t) - D(X_t) \tag{2.3}$$

where $V(X_t)$ is the production and $D(X_t)$ the degradation term. The corresponding chemical Langevin equation (CLE)[65] we use to model molecular noise in transcription and translation processes is

$$\frac{dX_t}{dt} = V(X_t) - D(X_t) + c\left(\sqrt{V(X_t)}\,\eta_v + \sqrt{D(X_t)}\,\eta_d\right) \tag{2.4}$$

where η_v and η_d are independent Gaussian white-noise processes[65]. c is a multiplicative constant to control the amplitude of the molecular noise. For each gene i, we use the Stratonovich scheme and the Milstein method to integrate two equations of the form of (2.4), one describing the rate of change of mRNA concentration and one for the rate of change of protein concentration (see **Appendix E**).

This model is derived from stochastic kinetics and the underlying assumptions are discussed by[65]. Note that, according to this model, a gene that is not activated ($V(X_t)$ close to zero) has a very low level of noise (leakage) and it can not suddenly have a very high transcription rate due to noise. In contrast, a gene that is activated has a higher level of noise (which may be interpreted as transcriptional bursts, for instance).

The measurement noise depends on the technology used to monitor gene expression concentrations[54] and is modeled here independently of the molecular noise. GNW implements Gaussian and log-normal models of experimental noise as well as a model of noise observed in microarrays[54].

2.2.3 Synthetic expression datasets

The next step in generating *in silico* benchmark networks consists in simulating the generated *in silico* regulatory networks to produce synthetic gene expression datasets. Available experiments in GNW are

- **Wild type.** The steady-state levels of the wild type (the unperturbed network).
- **Knockout (null-mutant).** Steady-state levels of single-gene knockouts (deletions). An independent knockout is provided for every gene of the network. A knockout experiment is simulated by setting the transcription rate of this gene to zero.
- **Knockdowns (heterozygous).** Steady-state levels of single-gene knockdowns. A knockdown of every gene of the network is simulated. Knockdowns are obtained by reducing the transcription rate of the corresponding gene by half.
- **Dual knockouts.** Dual knockouts consist of simulating a network with two genes knocked out simultaneously.
- **Multifactorial.** Steady-state levels of variations of the network, which are obtained by applying multifactorial perturbations to the network. One may think of each experiment as a gene expression profile from a different patient, for example. We simulate multifactorial perturbations by increasing or decreasing the basal activation of all genes of the network simultaneously by different random amounts (**Fig. 2.4**).

Custom perturbations can also be specified. Experiments can be simulated as steady states and/or time series with user-defined duration and number of measurement points.

2.3 Performance profiling of network inference methods

2.3.1 Evaluation of network inference methods

We not only provide researchers with a method for generating *in silico* gene network models to be used as benchmarks for reverse engineering algorithms, but also tools to facilitate the evaluation of network predictions. From a set of predictions from one or several inference methods, GNW automatically generates a comprehensive report including the result of a network motif analysis, where the performance of inference methods is profiled on local

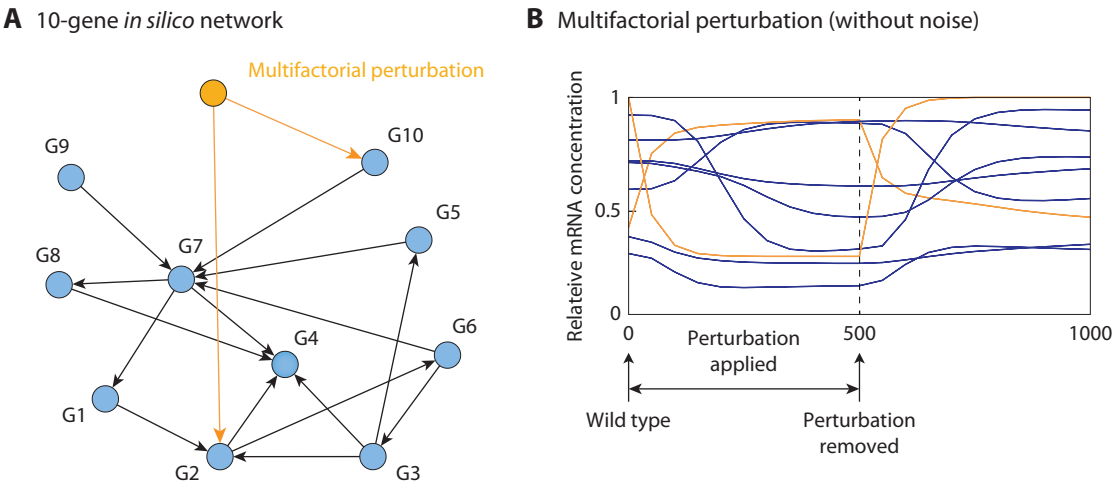


Figure 2.4: Multifactorial perturbation of a 10-gene *in silico* network. **(A)** Structure of the *in silico* network extracted from *E. coli* transcriptional network available in GNW. **(B)** Simulations of the dynamics of the network can be performed either deterministically or stochastically to account for molecular noise. Different types of experiments are also available in GNW including knockout, knockdown, an multifactorial perturbations to produce steady-state and time-series gene expression data. Here we display the time series of the mRNA concentrations when a multifactorial perturbation is applied to the gene G2 and G10. In order to produce more informative time series, the perturbation is released at the middle of the experiment.

connectivity patterns. The network motif analysis often reveals systematic prediction errors, thereby indicating potential ways of network reconstruction improvements[62]. Furthermore, precision-recall (PR) and receiver operating characteristic (ROC) curves are evaluated for each network prediction[66]. The relation between ROC and PR curves is discussed by Davis *et al.*[67].

We assessed the performance of six inference methods to illustrate benchmarking and performance profiling of network inference methods using GNW (**Table 2.1**). We first describe how to generate suitable network benchmark suites for the testing of various hypotheses. Specifically, we designed benchmark suites to show how the performance of inference methods is affected by different sizes and structural properties of regulatory networks. In addition, we show how GNW can help to identify the most informative type of gene expression data that a given inference method could use to achieve the best possible reconstruction from *in vivo* experiments. Finally, we introduce the DREAM4 Network Inference Challenge we generated, which has been used to assess the performance of many inference methods[68,69].

The intuitive interface of GNW allows to easily evaluate several inference methods at a time to facilitate the comparison of their relative performance. Evaluation results are always saved in a text file (XML format). In addition, GNW can generate PDF reports with plots from these data (an internet connection is required). Without internet connection, the evaluation can still be run but no PDF report will be created.

Table 2.1: Gene network inference methods evaluated using GNW. ARACNE2 and CLR are two of the most widely used inference methods. The following methods have been best-performer or co-best-performer in at least one DREAM challenge: Yip *et al.* (DREAM3 *in silico* Challenge Size 10, 50, and 100), Pinna *et al.* (DREAM4 *in silico* Challenge Size 100), and Huynh-Thu *et al.* (DREAM4 *in silico* Challenge multifactorial).

Inference method	Approach
ARACNE2[57]	mutual information (MI)
CLR[58]	mutual information (MI)
GENIE3[66]	regression
Z-score[66]	statistical
Pinna *et al.*[70]	statistical
Yip *et al.*[71]	noise model

Generation of network benchmark suites

We generated several network benchmark suites using the approach described in *Methods*. Each benchmark suite is composed of several *in silico* regulatory networks (the so-called *gold standards* or *target networks*). **Figure 2.5** shows one gold standard extracted from a regulatory network of the yeast *S. cerevisiae*. The extracted structures have been endowed with stochastic dynamical models of gene regulation accounting for molecular noise in transcription and translation processes.

The dynamical models of gene regulation have then been simulated to reproduce wild-type, knockout, knockdown, and multifactorial perturbation experiments. **Figure 2.6** illustrates the evolution of mRNA concentration levels without noise, when only molecular noise is introduced, and with both molecular and experimental noise. We generated the following benchmark suites:

- **Benchmark suite A.** 40 500-gene networks (20 from *E. coli* / 20 from yeast). Systematic knockout experiments were simulated to generate steady-state expression data.
- **Benchmark suite B.** 20 100-gene networks (10 from *E. coli* / 10 from yeast), 20 200-gene networks (10 from *E. coli* / 10 from yeast), and 20 500-gene networks (10 from *E. coli* / 10 from yeast). Systematic knockout experiments were simulated to generate steady-state expression data.
- **Benchmark suite C.** 20 100-gene networks (10 from *E. coli* / 10 from yeast). Systematic knockout and knockdown, and 100 multifactorial perturbation experiments were simulated to generate steady-state expression data.

At least half of the genes included in each gold standard are regulators, i.e. genes which regulate the mRNA production of at least one other gene. This is to avoid structures where

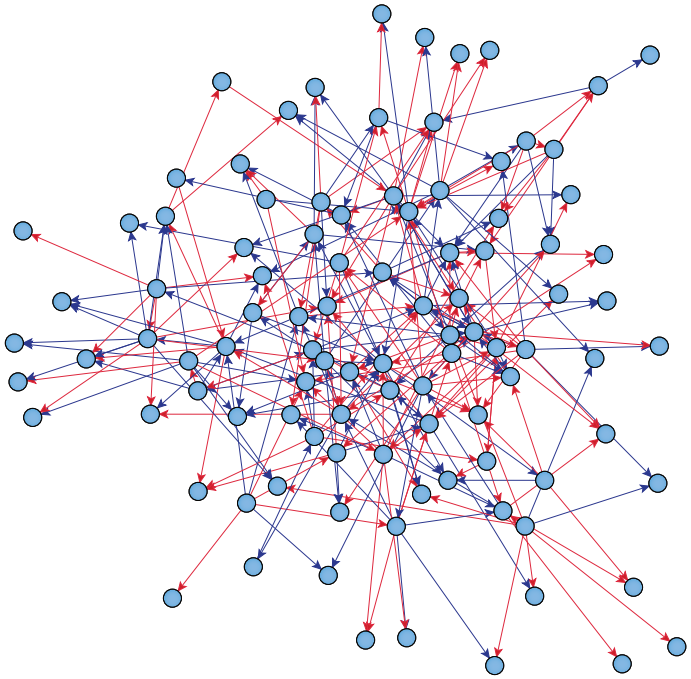


Figure 2.5: Fifty-gene *in silico* regulatory network extracted from the yeast *S. cerevisiae* transcriptional network using GNW. The evaluation of the performance of network inference methods requires the generation of many *in silico* benchmark networks of different sizes. Enhancing and inhibitory gene regulations are in blue and red, respectively. GNW also provides network visualization to quickly gain insight into the structure and dynamics of the network generated. Different layouts can be selected including Fruchterman-Reingold force-directed algorithm[72] which we use here.

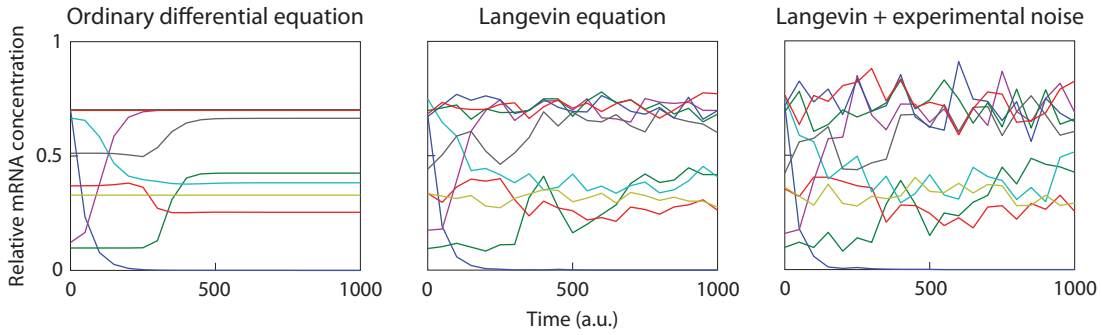


Figure 2.6: Simulation of the dynamics of *in silico* gene network models. Here we show the effect of both molecular and measurement noise on gene expression data. **(A)** The integration of the ODE model defined in (2.1) and (2.2) leads to noiseless gene expression. **(B)** Molecular noise is introduced by replacing equations (2.1) and (2.2) with stochastic differential equations (SDEs) defined in (2.4). **(C)** Superposition of both molecular and experimental noise.

there are many genes that do not regulate any other genes (out-degree = 0). We used the default parameter values proposed by GNW to simulate the gene expression experiments (see **Appendix B**).

2.3.2 Effect of network structural properties on inference method performance

The performance of network inference methods may strongly vary depending on the structural properties of the target networks. **Figure 2.7** shows systematic errors made by each inference method on four three-node motifs over-represented in the *in vivo* regulatory network structures of *E. coli* and yeast[14,73], and therefore in the gold standard structures we generated.

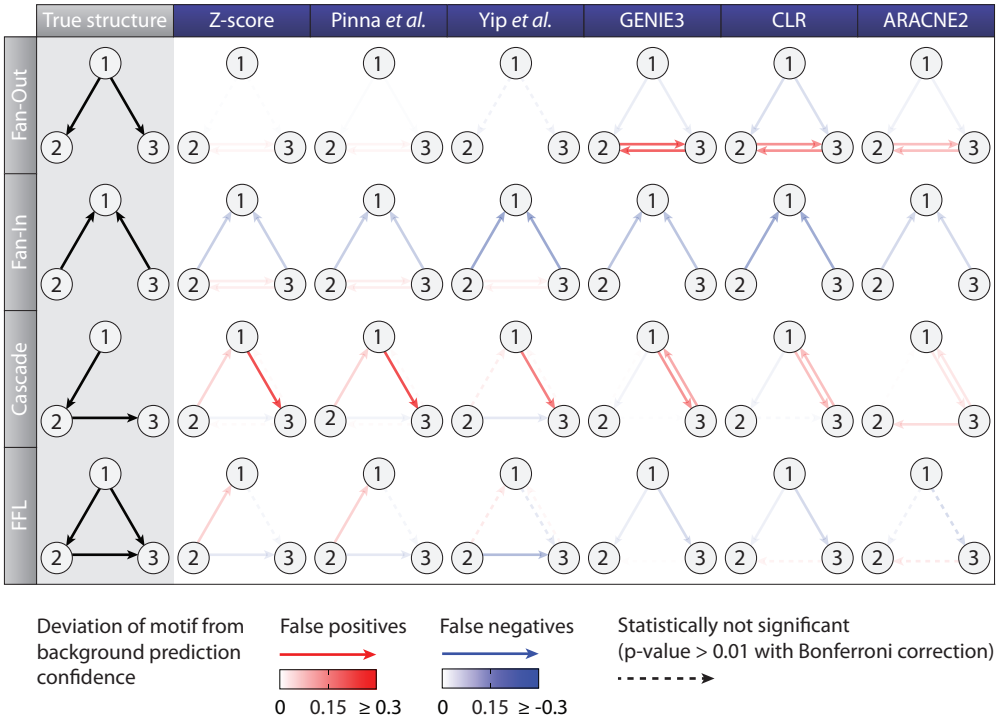


Figure 2.7: Systematic errors made by network inference methods in predicting network motifs. GNW analyses thirteen configurations of three-node motifs, including fan-out, fan-in, cascade, and feed-forward loop (FFL) motifs, which are over-represented motifs in *E. coli* and yeast regulatory network. The first column displays the network motifs to infer and additional columns show the systematic errors made by each inference method when trying to infer the corresponding network motif.

Z-score, Pinna *et al.*, and Yip *et al.* have different error profiles than CLR, ARACNE2 (both based on mutual information), and GENIE3, which make systematically false positive errors between gene 2 and 3 in predicting fan-out motifs. Note that ARACNE2 seems to make less errors on that particular motif because the gene interactions present in the gold standards

are in general less reliably identified than with CLR or GENIE3, independently of any network motifs considered. On the other hand, Z-score, Pinna *et al.*, and Yip *et al.* are strongly affected by cascade motifs, where these methods systematically predict false positive interactions between gene 1 and gene 3.

We show that inference methods have changing performance when used to make predictions about the structure of regulatory networks having specific structural properties. Thus we evaluated the selected inference methods (**Table 2.1**) against the benchmark suite *A* described in **Section 2.3.1**. **Figure 2.8** shows the AUROC and AUPR values obtained by those methods when applied to infer *E. coli* and yeast network structures from knockout expression data.

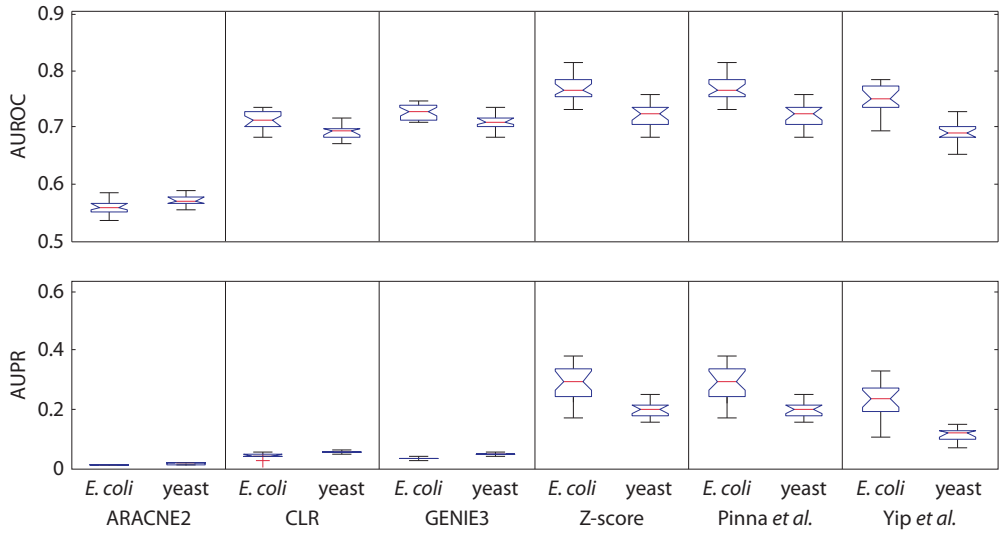


Figure 2.8: Effect of structural properties of target networks on performance of inference methods. 20 benchmark networks containing 500 genes each have been generated for each condition using GNW (benchmark suite *A*, see **Section 2.3.1**). The inference methods have been applied to predict the directed structure of each benchmark network from knockout expression data and the corresponding AUROC and AUPR values have been evaluated. Methods strongly impeded by the cascade motif (Z-score, Pinna *et al.*, and Yip *et al.*) as shown in **Figure 2.7** exhibit a performance degradation on yeast because yeast structure is composed of more cascade motifs than *E. coli* network structure.

The AUROC and AUPR values obtained by Z-score, Pinna *et al.*, and Yip *et al.* on yeast gold standards are significantly lower than on *E. coli* benchmark networks (Mann-Whitney U-test, $p < 0.01$). The performance degradation observed on yeast is due to the fact that these methods make systematic errors in predicting cascade motifs, and because structures extracted from yeast contain more cascade motifs than in *E. coli* structures (data not shown). We observe a linear correlation between the number of cascade motifs to predict in a regulatory network and the AUROC and AUPR values obtained for Z-score, Pinna *et al.*, and Yip *et al.* (Pearson's correlation, $-0.703 \leq r \leq -0.552$, $p < 0.05$). ARACNE2, CLR, and GENIE3 are less affected by the cascade motif (**Fig. 2.7**).

Interestingly, **Figure 2.7** also shows that Z-score and Pinna *et al.* exhibit very similar error profiles. Z-score is one of the simplest inference methods[66], yet it has relatively high accuracy in predicting network structures from knockout steady states. Pinna *et al.* first performs a Z-score analysis followed by a refinement stage, which aims to suppress the errors made by Z-score on cascade motifs[70]. **Figure 2.7** does not show any noticeable difference between Z-score and Pinna *et al.* This is confirmed by the fact that AUROC and AUPR values for Z-score and Pinna *et al.* are not significantly different (Mann-Whitney U-test, $p > 0.05$).

2.3.3 Effect of network size on inference method performance

We are interested in showing how the performances of inference methods scale with the size of the regulatory networks to reconstruct. Using GNW, it is very simple to generate *in silico* benchmark network of size $N < M$, where M is the size of the source network used (e.g. *E. coli* or yeast). Here we used the benchmark suite *B* described in **Section 2.3.1**, where each benchmark network has been simulated using the above methodology to produce knockout gene expression data. **Figure 2.9** shows the performance of the inference methods listed in **Table 2.1** when applied to infer regulatory networks containing 100, 200, and 500 genes.

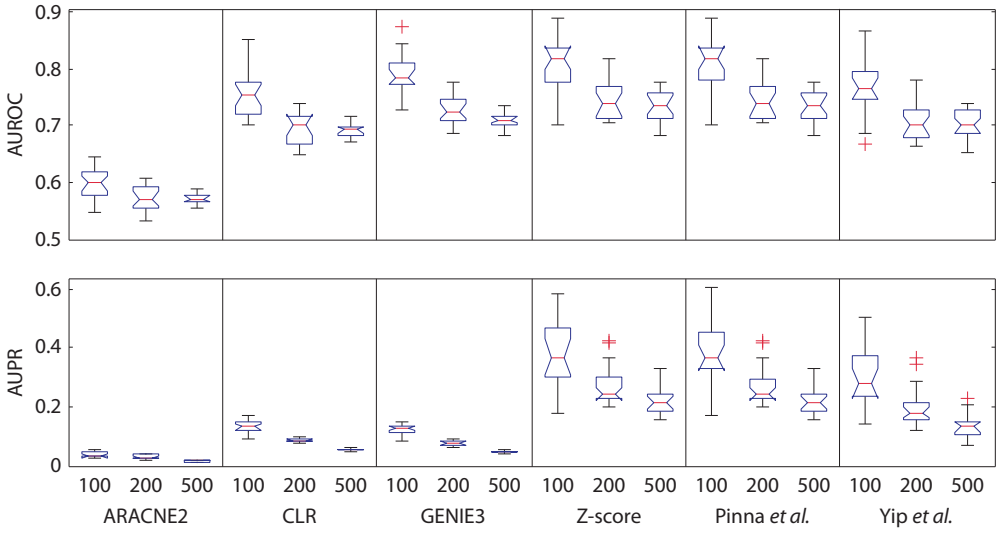


Figure 2.9: Performance assessment of inference methods on GNW-generated *in silico* benchmark networks of size 100, 200, and 500 genes. 20 benchmark networks have been generated for each condition (benchmark suite *B*, see **Section 2.3.1**). The inference methods have been applied to predict the directed structures of benchmark networks from knockout expression data and the corresponding AUROC and AUPR values have been evaluated. We observed that the performance of inference methods decreases with the size of the regulatory networks to reconstruct.

CLR has both AUROC and AUPR values significantly higher than those obtained by ARACNE2 for gold standards of size 100, 200, and 500 (Mann-Whitney U-test, $p < 0.01$). Leaving

ARACNE2 aside, AUROC values of the five remaining methods are comparable. However, we identified three methods with relatively high AUPR values. They are Z-score, and the methods developed by Pinna *et al.* and Yip *et al.* AUROC and AUPR values obtained by Z-score and Pinna *et al.* are significantly higher than those of Yip *et al.*, and this is valid for every gold standard size (Mann-Whitney U-test, $p < 0.05$). Also, Z-score, Pinna *et al.*, and Yip *et al.* have high AUPR variances because they are strongly affected by cascade motifs (see **Fig. 2.7**), which are more frequent in gold standards extracted from yeast than *E. coli* (each condition in benchmark suite *B* is composed of 20 gold standards, half being extracted from *E. coli* and half from yeast).

Figure 2.9 shows that the AUPR values of inference methods decreases as the sizes of the gold standards increase. The reason is that the connectivity density of the regulatory networks is higher for smaller networks. The higher the connectivity density, the easier it is for each of the six inference methods to have a high AUPR value (Pearson's correlation, $0.383 \leq r \leq 0.839$, $p < 0.01$).

2.3.4 Design of *in vivo* gene expression experiments

A given inference method may require a very specific type of expression data in order to enable accurate network reconstruction. We show that *in silico* benchmark networks have also the ability to support the design of suitable *in vivo* gene expression experiments, which are typically time-consuming and expensive[19]. The benchmark suite *C* described in **Section 2.3.1** is formed of 20 *in silico* networks consisting of 100 genes each, which we simulated using GNW to produce steady-state data for systematic knockout and knockdown, as well as 100 multifactorial perturbation experiments. **Figure 2.10** shows the AUROC and AUPR values obtained by the inference methods reviewed here (**Table 2.1**).

The most accurate network reconstructions are obtained using GENIE3, Z-score, and the methods developed by Pinna *et al.* and Yip *et al.* on knockout data. Knockout experiments are very informative because they provide network responses to individual and large perturbations (genes are "deleted"). Knockdown expression data, where the maximum transcription rate of genes is halved, are less informative than knockout data and thus lead to less accurate network reconstructions. **Figure 2.10** shows that ARACNE2 obtained AUROC and AUPR values comparable to CLR and GENIE3 when using multifactorial perturbation data. In addition, we considered providing knockout, knockdown, and multifactorial perturbation data together to ARACNE2, CLR, and GENIE3. We observed that AUROC and AUPR values obtained were slightly higher than when providing individually the three expression datasets (data not shown). We also added successively 100, 200, 300, and 400 additional multifactorial perturbations, however, the AUROC and AUPR values didn't improve significantly for all methods (Mann-Whitney U-test, $p < 0.05$). Furthermore, it has been shown using GNW and time-series data that the inference accuracy of inference methods reaches a saturation point after a specific data size[74]. This reveals that simply adding more expression data does not necessarily imply

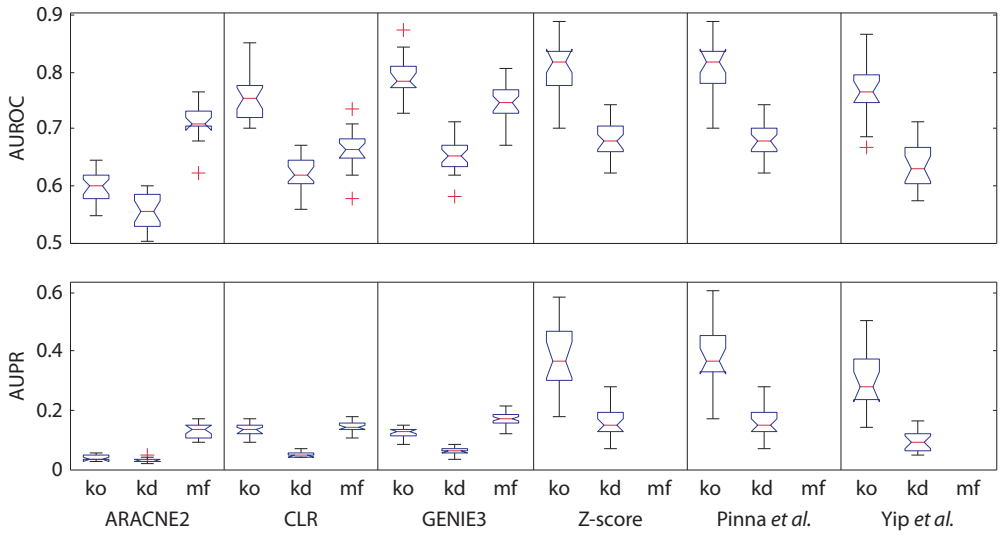


Figure 2.10: Identification of the most informative type of gene expression data required by a given inference method using *in silico* benchmark networks. Knockout (ko), knockdown (kd), and multifactorial (mf) perturbations were applied on 20 gold standards to generate three datasets containing each 100 measured steady states (benchmark suite *C*, see **Section 2.3.1**). Note, Z-score, Pinna *et al.*, and Yip *et al.* are not applicable to the multifactorial data.

performance improvement.

2.3.5 DREAM Network Inference Challenges

We have used GNW to generate the target networks for three international competitions on gene network reverse engineering: DREAM3 (2008), DREAM4 (2009), and DREAM5 (2010). Participants of the DREAM4 *in silico* Challenge were asked to provide network predictions for two sub-challenges made of networks of size 10 and 100, respectively. Each sub-challenge was composed of five *in silico* gene networks (two extracted from *E. coli* and three from yeast), which have been simulated to produce steady-state wild-type, knockout, knockdown, and multifactorial perturbation experiments. In addition, time-series data have been made available.

For each sub-challenge, network predictions made by participating teams have been evaluated by computing P-values, which indicate the probability that random lists of genetic interaction predictions would be of the same or better quality[66]. The overall score that has been used for ranking of the methods applied in the DREAM4 *in silico* Challenge was a negative log-transformed P-value given by

$$\text{overall score (OS)} = -0.5 \cdot \log_{10}(p_1 p_2) \tag{2.5}$$

where p_1 and p_2 are respectively the geometric means of AUPR P-values and AUROC P-values taken over the five networks. Thus, larger scores indicate smaller P-values, hence better predictions. **Figure 2.11** compares the overall scores of the inference methods reviewed here (**Table 2.1**) to those obtained by the participating methods applied in the DREAM4 *in silico* Size 100 Challenge.

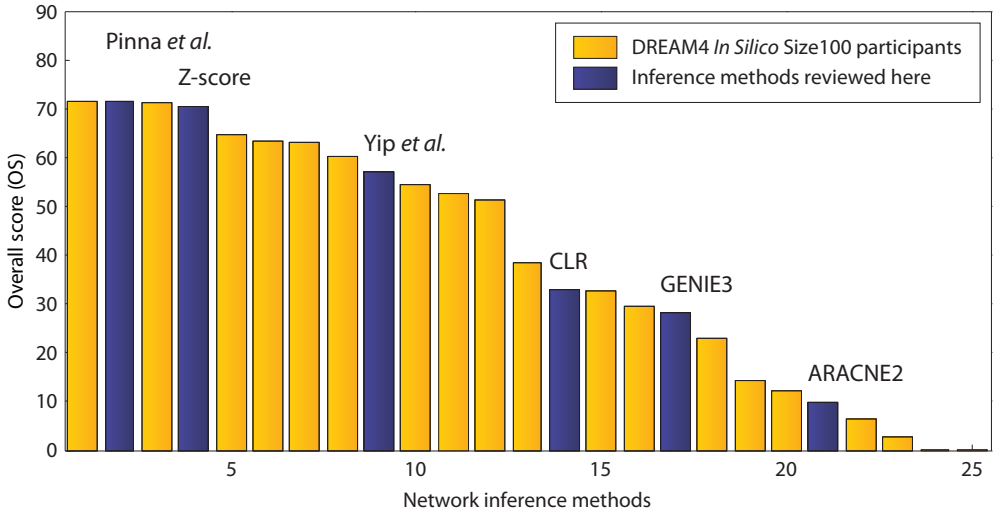


Figure 2.11: Performance assessment of inference methods listed in Table 2.1 on the DREAM4 *In Silico* Size 100 Challenge. Methods are ranked according to the geometric means of AUPR P-values and AUROC P-values taken over five networks. Pinna *et al.* was best-performer in that challenge, hence the two first bars correspond both to the overall score of Pinna *et al.* Typically, inference methods accept different types of gene expression data as input. Each method reviewed here has been fed with the maximum amount of accepted expression data.

The most accurate reconstruction of the five gene networks of size 100 genes was achieved by[70]. They participated to the DREAM4 *in silico* Size 100 Challenge, in which their method was *best-performer* (OS = 71.589). Hence, both first bars in **Figure 2.11** correspond to the score of Pinna *et al.* We have shown in **Figure 2.7** that AUROC and AUPR values obtained by Pinna *et al.* are not significantly higher than those obtained using the original Z-score method. This can be explained by the fact that transitive causal effects are almost always weaker than the direct effects. We expect that if many amplifying cascades occur, the refinement stage introduced by[70] will enable more reliable network predictions as compared to Z-score alone.

It is also interesting to note that the method of Yip *et al.* has been best-performer on all DREAM3 *in silico* Challenges of size 10, 50, and 100 genes we also provided. Yet it would have been ranked 7^{th} on the DREAM4 size 100 challenge (OS = 57.079). While the original algorithm is composed of several batches using both steady-state and time-series data, Yip *et al.* only used the first batch to build a noise model from knockout steady-state data[71]. The achievement of the 7^{th} rank in DREAM4 can be partially explained by the fact that Yip

et al. made a strong and correct assumption on the Gaussian measurement noise we used in DREAM3, which is no longer valid in DREAM4. Indeed, we modeled molecular noise in addition to a model of experimental noise observed in microarrays[54].

2.4 Conclusions

We propose a comprehensive and powerful framework for *in silico* benchmark generation and performance profiling of network inference methods. We implemented this framework as an open-source tool called GeneNetWeaver (GNW). Biologically plausible network structures are generated by extracting modules from known biological interaction networks such as those of *E. coli* and the yeast *S. cerevisiae*. Network structures are then endowed with detailed dynamical models of gene regulation describing both transcription and translation processes. Transcriptional regulation is modeled using a thermodynamic approach accounting for both independent ("additive") and synergistic ("multiplicative") interactions. In addition, our models account for stochastic molecular noise as well as experimental noise observed in microarrays. The generated *in silico* benchmark networks can be simulated in GNW to reproduce wild-type, knockout (null-mutant), knockdown (heterozygous), and multifactorial perturbation gene expression experiments. As an example of the application, we have used GNW to generate the target networks for three international competitions on gene network reverse engineering: DREAM3 (2008), DREAM4 (2009), and DREAM5 (2010). In total, 91 teams have submitted over 900 network predictions on GNW-generated networks, making GNW one of the most widely used benchmark generators by the community.

In contrast to previously proposed benchmark generators, GNW also integrates tools for systematic evaluation of the predictions from inference methods on benchmark networks. A unique feature of GNW is the ability to perform a network motif analysis from a set of network predictions and their corresponding benchmark networks. The network motif analysis reveals systematic prediction errors made by inference method on specific network motifs, thereby indicating potential ways of network reconstruction improvements. The accuracy of network inference is assessed using standard metrics such as precision-recall and receiver operating characteristic (ROC) curves.

We have used GNW to generate *in silico* benchmark suites to assess the performance and identify the strengths and weaknesses of six network inference methods. We show that Z-score, and the inference methods developed by Pinna *et al.* and Yip *et al.* make more accurate network predictions than two widely used methods, ARACNE2 and CLR. This good performance is achieved apparently because those methods target the inference of causal relationships between genes. Moreover, ARACNE2, CLR, and GENIE3 methods can be applied to infer regulatory networks even if no systematic knockout or knockdown experiments are provided (systematic knockout or knockdown experiments are typically not always available in practice). Furthermore, our results show that at some point simply giving more expression data to inference methods does not necessarily imply performance improvement. Therefore, the inte-

gration of additional information about the target regulatory networks should be considered, for instance using prior knowledge about the network structures.

The novelty of GNW is that it additionally provides a unique network motif analysis, which we used to show that the structural properties of the target regulatory networks affect the performance of inference methods. We observed that the performances of Z-score, and the methods developed by Pinna *et al.* and Yip *et al.* are impeded by the presence of cascade motifs in the target networks. Thus, we show that those methods make significantly less accurate network predictions on the yeast *S. cerevisiae*, whose structure includes more cascade motifs than *E. coli* transcriptional network structure. Finally, we also provide evidence that *in silico* benchmark networks can be used to identify the most informative type of gene expression data that a given inference method could use to achieve the best possible reconstruction from *in vivo* experiments.

3 Extensible and modular community detection in networks

Biological interaction networks are often organized into groups or modules of related genes and proteins carrying out specific biological functions. Over the past few years, many methods have been proposed to rationally decompose biological networks into functional modules, yet this challenging problem has not yet been successfully solved.

Here we introduce *Jmod*[a], an extensible and open source software for community structure detection in complex networks. A detection is first performed by applying one of the state-of-the-art module detection methods implemented in Jmod including Newman's spectral algorithm and a genetic algorithm-based method we developed. The accuracy of the module inference can then be further improved using one or several refinement methods independently of the detection method applied. The principal advantage of this modular framework is that different module detection methods and refinement techniques can be selected depending on the desired trade-off between speed and inference accuracy. We evaluate the performance of the module detection methods and refinement algorithms using biological and *in silico* networks. Because the methods considered here are based on modularity optimization, we discuss how their performance is affected by the *resolution limit* which is known to affect similar methods. Finally, we propose a voting method to combine multiple partitions generated by our GA-based detection method to overcome the resolution limit and so enable more robust and reliable network module inference.

[a] tschaffter.ch/projects/jmod

3.1 Introduction

Many systems can be described as complex networks where nodes represent individual units connected by edges depending on their relative interaction or relationship. The structure of a complex network has features that do not occur in random networks but often occur in real graphs. Examples include metabolic, protein and genetic interaction networks as well as neural, social and technological networks[20–23].

In such networks, there are usually groups of nodes that are more highly connected to each other than to the rest of the network. Such groups are usually called *modules*, *communities* or *clusters*[20,24]. In social networks, nodes often represent people connected by friendship relations and detecting communities (groups of friends or people with the same interest, family, etc.) have numerous applications[20]. In protein-protein interaction (PPI) networks, there are often groups of proteins that interact tightly in a same complex[6,21]. Therefore, the identification of these modules is of great importance to unravel the structural and functional properties of these networks.

Over the past few years, significant efforts have been made to develop methods for inferring the community structure of complex networks. Many techniques have been proposed including modularity optimization, fast greedy algorithms, mathematical programming or simulated annealing (see [75,76] for reviews). Despite this effort, the detection of modules in networks remains a very difficult task which has not yet been successfully addressed[25].

Here
interaction networks which we make available as an open source software called Jmod[b]. The inference of a network community structure is initially performed using one of the state-of-the-art module detection methods available in Jmod including Newman's spectral algorithm and a genetic algorithm-based method we developed. The obtained partition of a network into modules can then be improved using one or several refinement techniques which can be selected independently of the module detection method applied (**Fig. 3.1**). Here we introduce two refinement methods, the *Moving Vertex Method (MVM)* and the *global Moving Vertex Method (gMVM)* which we developed. This modular framework for community structure detection can be extended with additional community structure detection methods and refinement techniques to further increase its modularity. Moreover, different methods can be selected to address different speed-accuracy trade-off, hence allowing to target many different applications.

In **Section 3.2**, we describe three modularity optimization methods including Newman's spectral algorithm[27], a genetic algorithm-based method, and a brute force method. In **Section 3.3**, we introduce the refinement methods MVM and gMVM. In **Section 3.4.1**, we generate synthetic or *in silico* benchmark networks where the identity of the communities is known (the *ground truth*)[25]. This benchmark is later used to evaluate the performance of the module

[b] tschaffter.ch/projects/jmod

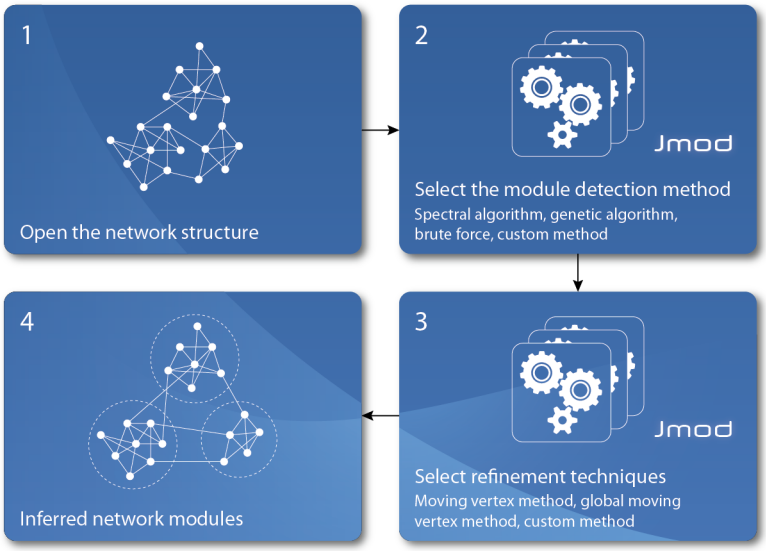


Figure 3.1: Extensible and modular community structure detection in networks. The input is a complex network whose nodes represent individual units connected by edges depending on their relative interaction or relationship. A first partition of a network into module is obtained by applying one of the module detection methods implemented in Jmod. The accuracy of the inference can then be further improved by applying different refinement methods independently of the module detection method selected. This modular method for community structure detection is available as an extensible ,open source Java software called Jmod.

detection methods previously introduced. In **Section 3.4.3**, we evaluate the effects of the genetic algorithm parameters on the performance of our module detection method. In the following sections, we profile the performance of each module detection methods and refinement methods using *in silico* and real social and biological networks. Finally, we discuss in **Section 3.4.6** the *resolution limit*[77] of the above modularity optimization methods described in this chapter.

3.2 Module detection methods

3.2.1 Newman's spectral algorithm

A great challenge in community structure detection is how to determine the performance of a module inference, that is, how to measure the quality of the partition of a network into modules. Newman and Girvan addressed this question by defining a quantity called *modularity Q* to evaluate the partition of a set of nodes into modules or communities[78]. Newman later proposed a slightly different definition of the modularity which can be summarized as

Q =(fraction of edges falling within modules)
- (expected fraction of such edges in randomized graphs)

Modularity compares the number of edges within each community detected with the expected number of edges in a random network of the same size and same distribution of node degrees[27]. The problem of finding the community structure in complex networks becomes an optimization problem whose solution corresponds to the partition of a network that maximize Q. The formal definition of Q given by Newman is

$$Q = \frac{1}{4m}\mathbf{s}^T\mathbf{B}\mathbf{s} \tag{3.1}$$

where n and m are respectively the total number of nodes and edges in the network and **B** is the so-called real symmetric *modularity matrix*[27] whose elements are given by

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m} \tag{3.2}$$

A_{ij} is the number of edges between the nodes i and j whose node degree is k_i and k_j. As a reminder, the node degree correspond to the number of edges attached to a given node. Because undirected structures are here considered, the adjacency matrix **A** and therefore **B** are symmetric.

The remaining term in (3.1) is the so-called *split vector* **s**. In Newman's algorithm, the community structure of a network is found by recursively splitting communities in two subcommunities. The network is initially split in two subnetworks or subcommunities. The split is defined by the n-by-1 vector **s** whose elements take the value $s_i = 1$ or $s_i = -1$ if the node i must be placed in the first or second subcommunity. The objective of the community structure detection method is thus to find the vector **s** that maximizes Q when inserted in (3.1).

Newman's method is termed as *spectral algorithm* because the bi-partitioning of a set of nodes in two subcommunities is obtained from the eigendecomposition of the modularity matrix **B**[27] so that **s** is expressed as a linear combination of its eigenvectors v_i in

$$Q = \frac{1}{4m}\sum_{i=1}^{n}\left(v_i^T\mathbf{s}\right)^2\lambda_i \tag{3.3}$$

$$\approx \frac{1}{4m}\left(v_1^T\mathbf{s}\right)^2\lambda_1 \tag{3.4}$$

where λ_i is the eigenvalue associated to the eigenvector v_i and $|\lambda_i| > |\lambda_{i+1}|$ for $i = 1,2,...,n-1$. In (3.4), Q is approximated by using only the *leading* eigenvector v_1 associated to the largest (most positive) eigenvalue λ_1[27]. Detailed information about how to compute the eigenpair

(λ_1, v_1) is given in **Appendix C**. The split of a community in two subcommunities is then obtained by selecting **s** as parallel as possible to the eigenvector v_1 so that

$$s_i = \begin{cases} 1 & \text{if } v_{1,i} > 0 \\ -1 & \text{otherwise} \end{cases} \tag{3.5}$$

After splitting the network in two subnetworks, the above spectral method is applied to further split each subnetwork in two subcommunities and so on (**Fig. 3.2**). After the first split of the network, the modularity matrix **B** is replaced in (3.1) by the *generalized modularity matrix* $\mathbf{B}^{(g)}$ to compute additional contributions ΔQ later added to the overall modularity Q of the network. For the second and subsequent splits, (3.1) is replaced by

$$\Delta Q = \frac{1}{4m}\mathbf{s}^T\mathbf{B}^{(g)}\mathbf{s} \tag{3.6}$$

where the elements of $\mathbf{B}^{(g)}$ are adapted from the elements of **B** as follows

$$B_{ij}^{(g)} = B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik} \tag{3.7}$$

δ_{ij} is the Kronecker δ-symbol, which takes the value 1 if $i = j$, otherwise 0. g corresponds to the group of nodes that are being split in two.

Furthermore, a community is *indivisible* if splitting it in two does not contribute positively to the overall modularity Q. Formally, a community is indivisible if there is no split vectors **s** for a group of nodes that lead to $\Delta Q > 0$ (or $Q > 0$ for the first split of the network). It is interesting to note that Q can be negative when the number of edges within the communities do not exceeds the number of such edges in randomized graphs. Q and ΔQ can take values in $[-1/2, 1)$ with large positive values of Q corresponding highly modular networks[78].

3.2.2 Genetic algorithm-based method

Newman's spectral algorithm represents the modularity matrix **B** in terms of its eigenvalues and eigenvectors[27]. The vector **s**, which describes the split of a group of nodes in two subcommunities, is then selected to be as parallel as possible to the leading eigenvector v_1 according to (3.5). However, the additional information contained in the remaining eigenvectors of **B** is discarded and so only an approximation of Q is computed in (3.4).

Here we introduce a different community structure detection method based on the application of a genetic algorithm to find the optimal splits of communities in two subcommunities. Our approach relies on Newman's definition of the modularity Q[27], however we do not perform the eigendecomposition of **B** and so **s** is not selected to be parallel to its leading eigenvector.

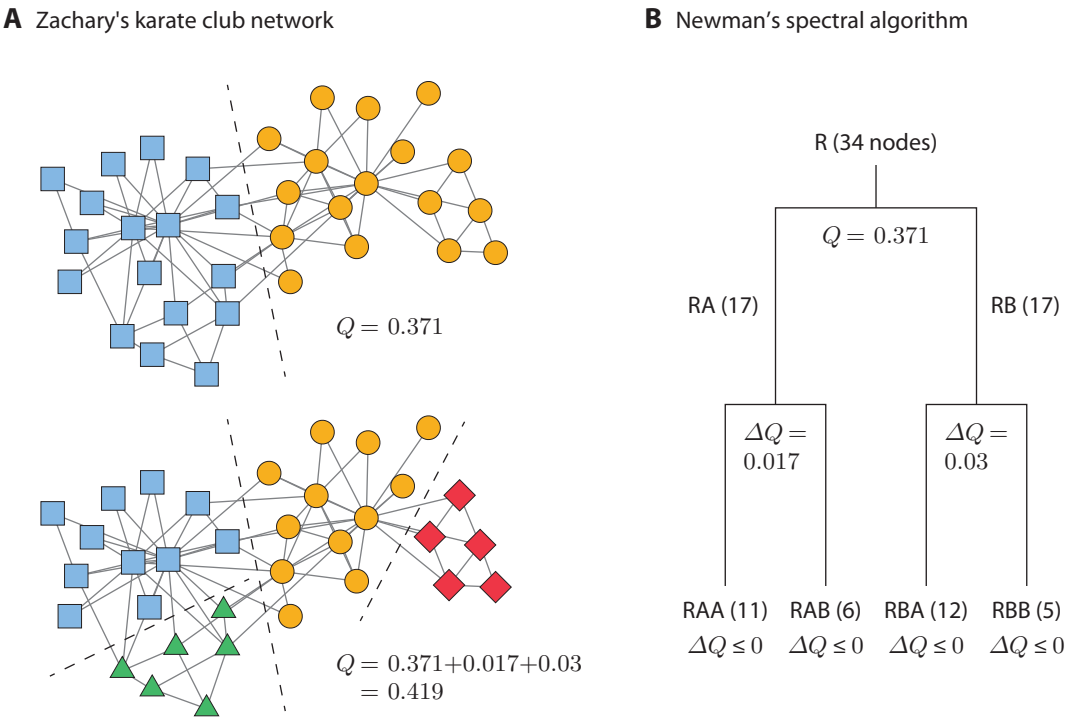


Figure 3.2: Community structure detection in Zachary's karate club network using Newman's spectral algorithm. (A) This network shows the friendship relations (edges) between the members of a karate club (nodes)[79]. An argument between the members eventually occurred, which led to the creation of a second club. As an example, we show that Newman's spectral algorithm can be used to predict the factions that would emerge if an argument between related people occurs. The first split of Zachary's network actually describes very accurately the composition of the two emerging clubs and only one person didn't join the expected faction[79]. **(B)** This dendrogram provides a visual representation of the history of the recursive community splits performed by Newman's algorithm. The size of each community is shown within parentheses and the leaves of the dendrogram represent the indivisible communities for which ΔQ is negative or equal to zero.

Instead, we apply a generational genetic algorithm (GA)[80,81] to evaluate many split vectors $\mathbf{s}$ before selecting the one that maximizes Q directly in (3.1). Hence, all the information contained in $\mathbf{B}$ is used and so Q is no longer approximated.

Genetic algorithms mimic the process of natural evolution to generate candidate solutions to optimization problems[80,82]. Initially, a population of candidate solutions or *individuals* is randomly generated. Here an individual represents a split vector $\mathbf{s}$ which describes the partition of a community in two subcommunities. $\mathbf{s}$ also corresponds to the *phenotype*. The *genotype* is the genetic representation of an individual which is later decoded to obtain the phenotype. Because the elements of $\mathbf{s}$ only take the values -1 and 1, we use a binary genetic encoding to represent these elements as 0 and 1 in the genotype of the individuals. Before

evaluating an individual, its genotype is thus decoded using the mapping $0 \rightarrow -1$ and $1 \rightarrow 1$.

The *fitness* of an individual is a scalar that indicates how good a solution is to a particular problem. Here the task is to optimally split a group of nodes in two subcommunities. The fitness of an individual is obtained by evaluating the expression of the modularity Q given by (3.1) as a function of $\mathbf{s}$ (the so-called *objective function* to maximize)[80].

Based on their fitness, individuals from the population are then selected for reproduction. The reproduction usually consists in combining the genetic material of two parents using crossover and mutation operators in order to produce new individuals or *offspring*, and so new candidate solutions (**Fig. 3.3**).

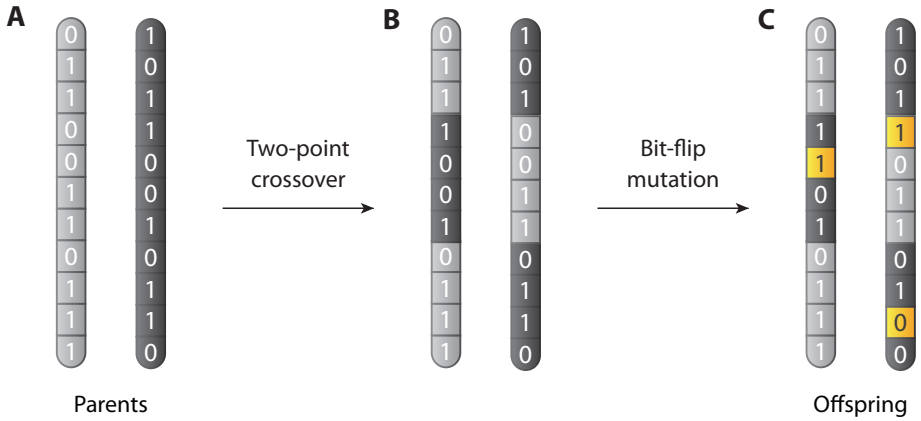


Figure 3.3: Reproduction of two individuals using crossover and mutation operators. (A) Two individuals are selected from the population for reproduction based on their fitness using tournament selection[83,84], for instance. **(B)** The crossover operator merges the genetic material from two parents to generate new individuals. **(C)** The mutation operator affects nucleotides under given probability to produce punctual changes in the genotype (in yellow). The amplitude of the changes depends on the encoding used, that is the relation between genotype and phenotype. Here a nucleotide corresponds to a single bit in the genotype which translates to $\{-1,1\}$ in the phenotype (the split vector $\mathbf{s}$). As $\mathbf{s}$ defines the split of a community in two subcommunities, mutating the nucleotide i results in flipping the sign of $\mathbf{s}_i$ and therefore in moving the node i from one subcommunity to other (**Section 3.2.1**).

After evaluation, offspring replace individuals in the main population under specific conditions[85]. The above algorithm is then repeated over many generations to produce and evaluate many individuals. Finally, the individual that has the largest fitness value in the final population is selected. In the present method for community structure detection in networks, the best split vector $\mathbf{s}$ found is applied to split the network in two subnetworks. Each subnetwork or subcommunity identified is further split in two using an additional GA run unless dividing one community does not further contribute positively to the overall modularity Q of the network (**Section 3.2.1**).

The performance of a GA depends on the values of several parameters including the size of the population, the genetic encoding used, the method used to select parents, and the crossover/mutation methods and rates[81,85]. The design and values of these methods and parameters depend on the *search space* to explore which is defined by the problem to solve and described by the fitness function. The identification of suitable GA parameters for our community structure detection method is discussed in **Section 3.4.3**.

3.2.3 Brute force method

The brute force method consists in evaluating every possible split vectors $\mathbf{s} \in \mathbb{R}^n$ before selecting the one that maximizes the modularity $Q(\mathbf{s})$ in (3.1). The total number of ways to split a n-node graph in two is 2^n. However, we observe that the two split vectors

$$\mathbf{s}_1 = [-1,-1,-1,\ \ 1,-1,\ \ 1,\ \ 1,-1,-1]$$
$$\mathbf{s}_2 = [\ \ 1,\ \ 1,\ \ 1,-1,\ \ 1,-1,-1,\ \ 1,\ \ 1]$$

actually define the same split so that both $\mathbf{s}_1$ and $\mathbf{s}_2$ obtain the same modularity value Q when they are evaluated in (3.1). There are finally $2^n/2 = 2^{n-1}$ distinct ways to split a n-node community in two subcommunities.

The advantage of the brute force method is that it guarantees to find the best possible split of a community in two, that is, to find the split vector $\mathbf{s}$ that maximizes the modularity $Q(\mathbf{s})$. However its application becomes quickly too computationally intensive as the total number of split vectors to evaluate doubles each time a network larger of one node is considered. Thus, the application of the brute force method remains limited to relatively small networks, typically with less than forty nodes (see **Appendix C.4**).

3.3 Refinement methods

In this section, we describe two algorithms designed to further improve the quality of community structures previously inferred using one of the methods described in **Section 3.2**. Both algorithms can be applied one after another for successive performance improvement.

3.3.1 Moving vertex method (MVM)

The moving vertex method (MVM) is the first technique used to refine a split vector $\mathbf{s}$ found using one of the community structure detection methods introduced in **Section 3.2** or any other modularity optimization methods[35,86].

The inspiration for this technique came from the Kernighan-Lin algorithm[87] which has been

initially proposed as a standalone technique for bi-partitioning graphs. It is important to note that Newman already used this technique as well as others[88,89] to improve the performance of his spectral algorithm[27], however without ever reporting or discussing its contribution to the modularity Q (see results in **Sections 3.4.4** and **3.4.5**).

MVM takes as input a split vector $\mathbf{s} \in \mathbb{R}^c$ that describes the split of c-node community in two subcommunities (**Section 3.2.1**). At each iteration, MVM generates and evaluates systematically c independent versions of $\mathbf{s}$ that differ by only one element at a time (**Fig. 3.4A**). Thus, flipping the sign of an element s_i translates into moving the node i from one to the other subcommunity before evaluating the effect of this modification on the Q in (3.1). The modification that leads to the largest increase in Q is then applied and the next iteration starts. Finally, the refinement procedure ends when no further modifications of $\mathbf{s}$ contributes positively to the modularity of the network (**Fig. 3.4B**).

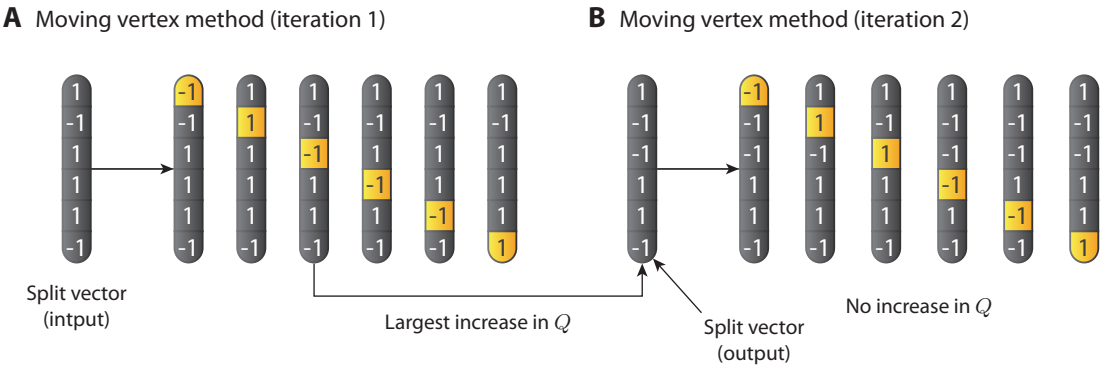


Figure 3.4: Refinement of the split of a community in two using the moving vertex method (MVM). (A) The split vector $\mathbf{s}$ returned by a community structure detection method is refined by systematically flipping the sign of only one element s_i, which translates into moving the node i from one to the other subcommunity. **(B)** The modified version of $\mathbf{s}$ that leads to the largest increase in Q is then selected and the next iteration starts. This optimization algorithm ends when there are no more $\mathbf{s}$ that improve Q.

The complexity of this refinement procedure has been evaluated to $O[(m+n)n]$ or $O\left(n^2\right)$ for a single split of a sparse graph where n and m are the number of nodes and edges[27]. Moreover, the procedure can be parallelized as the c modified versions of $\mathbf{s}$ are independent. The details of an efficient version of MVM as we implemented it in Jmod are given by **Algorithm C.2**.

3.3.2 Global moving vertex method (gMVM)

The refinement method MVM illustrated in the previous section is a relatively simple procedure that allows to improve the bi-partitioning of networks and subnetworks. However, its application remains limited to the splits of communities in two, thus localizing its effect to

only a portion of the network that is partitioned.

Here we introduce a second refinement technique called global moving vertex method (gMVM). This technique can be applied in addition to MVM as the final stage of any community structure detection methods to further improve the modularity Q of the network. As suggested by its name, gMVM takes inspiration from MVM. Instead of moving locally nodes between two subcommunities as MVM does, gMVM moves successively each node of the network to each indivisible community (**Section 3.2.1**). The move that leads to the largest increase in the overall modularity Q of the network is then applied. The new partition of the network is used as the starting point of the next iteration. The algorithm is repeated as long as there is a node move that increases Q. The details of the implementation of gMVM are given by **Algorithm C.3**.

Once again, the effect of MVM is limited to a portion of the network. However, gMVM can correct errors that may have been done all along the different stages of the community structure detection. gMVM can even redistribute all the nodes contained in a community misidentified as such among the remaining communities as long as the change improves the overall modularity Q of the network.

3.4 Evaluation of community structure detection methods

3.4.1 Generating Lancichinetti-Fortunato-Radicchi graphs

The evaluation of the performance of community structure detection methods is achieved here using benchmark graphs where the identity of the communities that compose them is known (also called *ground truth*).

Lancichinetti-Fortunato-Radicchi (LFR) benchmark graphs[90] are a generalization of the graphs proposed by Girvan & Newman[26] where all nodes have the same expected degree and all communities or modules have the same size. The node degrees in LFR graphs are distributed according to a power law with exponent τ_1. The community sizes follow a second power law distribution with exponent τ_2. The most interesting parameter in the generation of these benchmark graphs is the *mixing parameter* μ, which expresses the ratio between the external degree of a node with respect to its community and the total degree of the node[90]. The formal definition of μ is given by

$$\mu = \frac{k_i^{out}}{k_i^{in} + k_i^{out}} \tag{3.8}$$

where k_i^{in} and k_i^{out} are the internal and external degrees of node i with respect to its community. Expressed differently, k_i^{in} is the expected number of neighbors of i that belong to the same community and k_i^{out} the expected number of neighbors of i that belong to other

communities. From (3.4.1), a node then shares a fraction μ of its connections with nodes in other communities and a fraction $1-\mu$ of its connections with other nodes in its community. Therefore, the modularity of the benchmark graphs is inversely proportional to the value of the mixing parameter μ.

We follow the procedure described by Lancichinetti *et al.* to generate undirected and unweighted LFR benchmark graphs[90]. Especially, we generate benchmark graphs that include $n = 1000$ nodes with a node degree $< k >$ fixed to 20. The maximum node degree is set to 50. These parameter values actually match those used by Lancichinetti *et al.* in their comparative analysis of the performance of twelve community structure detection methods[25].

Moreover, we generate two additional classes of graphs. The first class of graphs is made of "small" communities including each between $c_{min} = 10$ and $c_{max} = 50$ nodes. The second type of graphs is composed of "big" communities including each between $c_{min} = 20$ and $c_{max} = 100$ nodes. The exponents of the power law distribution of the node degree τ_1 and community size τ_2 are set to -2 and -1, respectively[25]. Finally and for each set of the above parameters, we produce graphs that have different degrees of modularity by setting the mixing parameter μ to values in $[0.1, 0.9]$ with step size set to 0.05.

Figure 3.5 illustrates the generation of 1000-node LFR benchmark graphs that include small and big communities for $\mu = 0.1$ and $\mu = 0.5$. By painting the nodes of a few modules, we can observe that graphs generated for $\mu = 0.1$ are effectively more modular than those generated for $\mu = 0.5$ where communities are more mixed. According to the definition of given in (3.4.1), nodes for $\mu = 0.1$ share $9/10$ of their connections with other nodes of the same module. This ratio becomes $1/10$ for graphs generated for $\mu = 0.9$.

3.4.2 Evaluating the performance of module inference

Modularity Q

Modularity is the first quantity used to evaluate the quality of the partition of a network whose community structure is unknown[76,91]. The introduction of the concept of modularity opened the door to the development of numerous modularity optimization methods, whose goal is to find the partition of the network into modules that maximizes Q[75,76].

However, it has been demonstrated that modularity-based methods can make errors in predicting the community structure of networks including small communities. This is known as the *resolution limit* of modularity optimization methods[77]. In **Section 3.4.6**, we show how this limit affects the performance of the methods introduced at the beginning of this chapter. Especially, we show that the design of our GA-based module detection method makes it particularly robust to this resolution limit.

Another method is still required to evaluate the quality of network partitions. We decide to use

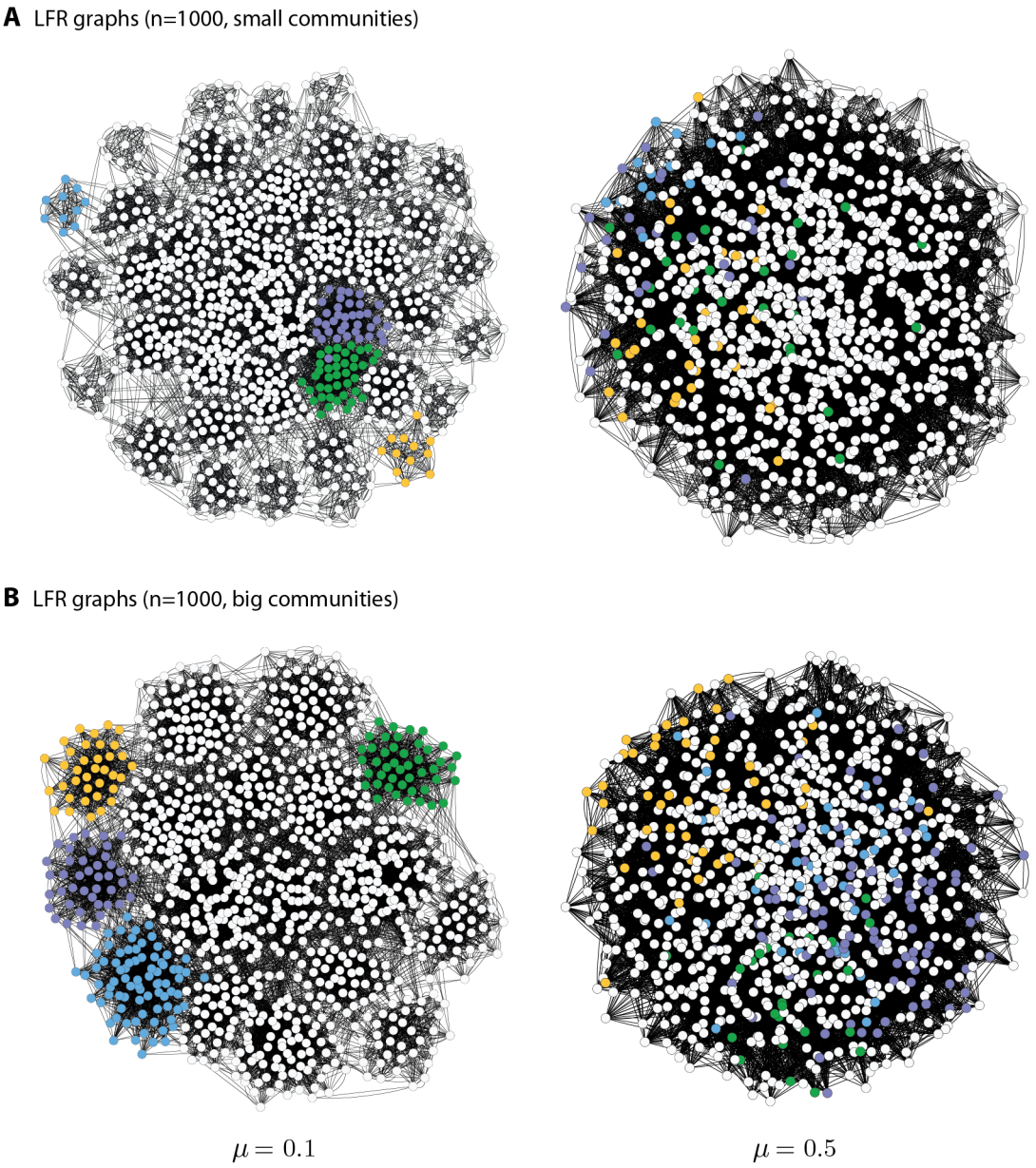


Figure 3.5: Structures of 1000-node Lancichinetti-Fortunato-Radicchi (LFR) benchmark graphs. These LFR benchmark graphs are composed of **(A)** small communities (10 to 50 nodes each) and **(B)** big communities (20 to 100 nodes each). The mixing parameter μ, which controls the ratio of external and internal node degrees, enables the generation of modular and less modular graphs[25]. Painting all the nodes of a few communities shows clearly that communities are more mixed in graphs generated for $\mu = 0.5$ than for $\mu = 0.1$. The network representations have been obtained using the *Force-Directed Layout* of Cytoscape[61].

the same quantity as the one used by Lancichinetti *et al.* in their comparative analysis[25] which makes use of the ground truth provided by their benchmark networks (see **Section 3.4.2**).

This will also allow us to directly compare the performance of our methods to those of other methods.

Normalized mutual information (NMI)

The similarity between two partitions of the same graph can be quantified by their *normalized mutual information (NMI)*[75]. The normalized mutual information is denoted $I(A,B)$ and is defined as

$$NMI(A,B) = \frac{-2\sum_{i=1}^{c_A}\sum_{j=1}^{c_B} n_{ij} log(\frac{n_{ij}n}{n_{i.}n_{.j}})}{\sum_{i=1}^{c_A} n_{i.} log(\frac{n_{i.}}{n}) + \sum_{j=1}^{c_B} n_{.j} log(\frac{n_{.j}}{n})} \tag{3.9}$$

where A is the partition inferred by the community structure detection method and B is the ground truth partition provided by the benchmark network, c_A and c_B is the number of modules in the partition A and B, and n is the number of node which is the same in the two partitions. n_{ij} is the number of nodes shared by the i^{th} module of A and the j^{th} module of B. $n_{i.}$ is the total number of nodes in the i^{th} module of A and $n_{.j}$ is the total number of nodes in the j^{th} module of B. Moreover, the convention $0 \cdot log(0) = 0$ is used for the calculation of (3.9).

$NMI(A,B)$ takes values in $[0,1]$, where 0 is obtained when the partitions A and B are totally independent and 1 when the two partitions that are identical. Several comparative analyses have alread used this pairwise measure of similarity to compare the performance of community structure detection methods[25,75].

3.4.3 Evaluation of the genetic algorithm parameters

The performance of the GA-based community structure detection method introduced in **Section 3.2.2** depends on several methods and parameters such as the genetic encoding used, the fitness function to maximize and the genetic operators applied to produce new candidate solutions[81,85]. In the following sections, we benchmark the GA parameters to provide insight into how each of them affect the accuracy of the community structure inference.

Genetic representation and fitness function

We use a binary encoding to represent the GA individuals, which represent split vectors $\mathbf{s} \in \mathbb{R}^c$ where c is the size of the community being split in two subcommunities (**Section 3.2.2**). The initial population of the GA is composed of 100 random individuals. The genome of each individual in the population is then decoded and evaluated using the fitness function $Q(\mathbf{s})$. For the first split of the network in two, $\mathbf{s} \in \mathbb{R}^n$ where n is the size of the network and the fitness function $Q(\mathbf{s})$ defined by (3.1) is used. For subsequent splits of the communities, the fitness function evaluated is the additional contribution to modularity $\Delta Q(\mathbf{s})$ defined by (3.6)[27].

Selection and reproduction

The selection of parents for reproduction is performed using tournament selection with tournament size set to two[84]. The tournament selection picks randomly two individuals from the population before selecting the one with the largest fitness as parent for reproduction. Increased selective pressure could be obtained by increasing the size of the tournament selection, however this would be achieved to the detriment of the population diversity, which is required to efficiently explore the search space of solutions[84].

We enable elitism selection so that the best individual in the current generation is carried over unaltered to the next generation. The number of elites conserved at each generation is set to one. As for increasing the size of the tournament selection, using more than one elite would lead to a decrease in the population diversity[81].

The generation of new individuals or offspring is achieved by selecting two parents from the population using tournament selection before combining their genetic material using crossover and mutation operators[85,92]. This reproduction procedure is then repeated to generate one hundred individuals which then replace worse individuals in the population. The size of the population remains constant through the generations.

Crossover and mutation operators

The fruitful exploration of the search space by the GA to find the hopefully global optimal split of a community in two subcomunities relies heavily on the design of suitable genetic operators such as crossover and mutation operators[92]. In this section, we report the performance of different crossover and mutation methods when applied to identify communities in modular and non-modular LFR benchmark graphs.

The idea behind the crossover operator is that better offspring can be produced by combining the genetic material of two or more parents[93]. We consider here three crossover methods which are one-point, two-point and uniform crossovers[81,92]. One-point crossover splits the genome of two parents at a random point before swapping their genetic material. Because all individuals have the same genome length, which is given by the number of nodes to partition in two subcommunities, the location of the split point is the same for the two parents. Two-point crossover does the same but cuts the parents in three segments at two random points (**Fig. 3.3**). Unlike one- and two-point crossover, the uniform crossover exchanges the genetic material of two parents at the nucleotide level rather than at the segment level. The nucleotides correspond here to the binary elements $\{0,1\}$ that compose the genome of the individuals. Moreover, the application of the crossover operator to two parents is not systematic and depends on the selected *crossover rate*. In case the crossover operator is not applied, the genotype of one of the two parent is selected as it is.

The mutation operator is applied after crossover to randomly modify one or several nucleotides

of the genome with a given probability[93]. The bit-flip mutation operator is particularly suitable for binary representations and consists in flipping the value of a nucleotide ($0 \rightarrow 1$ or $1 \rightarrow 0$). In our application and when splitting a community in two, the value of the nucleotide i of an individual defines if the node i must be placed in the first or second subcommunity. Thus, flipping the value of the nucleotide or bit i translates into moving the node i to the other subcommunity. Similarly to the crossover operator, the *mutation rate* controls the rate at which the mutation operator is applied to modify the genotype of an individual. Here we define the mutation rate as the number of bits mutated in the genome of one individual.

The design of suitable genetic operators is required to efficiently explore the search space of all possible candidate solutions while enabling at the same time the convergence towards the hopefully global optimum. In order to identify the right balance between exploration and convergence, we profile the performance of different crossover and mutation methods and rates on different LFR benchmark graphs generated in **Section 3.4.1**.

Here we consider 1000-node graphs with small communities (10-50 nodes each). Moreover, we select graphs with different degree of modularity which is inversely proportional to the value of the mixing parameter μ. For each value of μ in $\{0.1, 0.5, 0.9\}$, we select one graph from the LFR benchmark described in **Section 3.4.1** and run the GA twenty times to perform the first split of the network in two subnetworks. **Figure 3.6** shows the average of the maximum modularity values $Q(\mathbf{s})$ achieved for different crossover and mutation methods over twenty repetitions.

Figure 3.6 shows that both crossover and mutation operators are required to find optimal splits of the networks in two. Without mutation, the GA fails to converge towards optimal solutions (dark vertical stripe on the left of each panel). This can be explain by the lack of diversity in the genetic material which makes the GA unable to explore the search space further than the closest local optimum at the time of the first generations. Moreover, the best performance of the GA is achieved for small amounts of mutation. Typically, one node mutated per genome allows the GA to find the optimal solutions for each graph, which will later appear to be the global optima (**Section 3.4.5**). The performance then decreases rapidly as the mutation rate increases. This is because large mutation rates prevent the GA to converge towards an optimum.

The three crossover methods considered achieves very similar performance. A general trend observed in modular as well as in non-modular graphs is that high crossover rates is required to achieve convergence before that the GA reaches the maximum number of generations, which is set here to 3000 generations. High crossover rates usually comes with a higher chance of premature convergence, which occurs when the GA is trapped on suboptimal solutions without being able to produce offspring better than their parents[81,85]. We show later that this risk is relatively low for networks with high and medium degree of modularity. Nevertheless, we describe in **Section 3.4.3** a strategy that we systematically apply to increase the chance of the GA to converge towards the optimal solution and thus circumvent the risk of premature

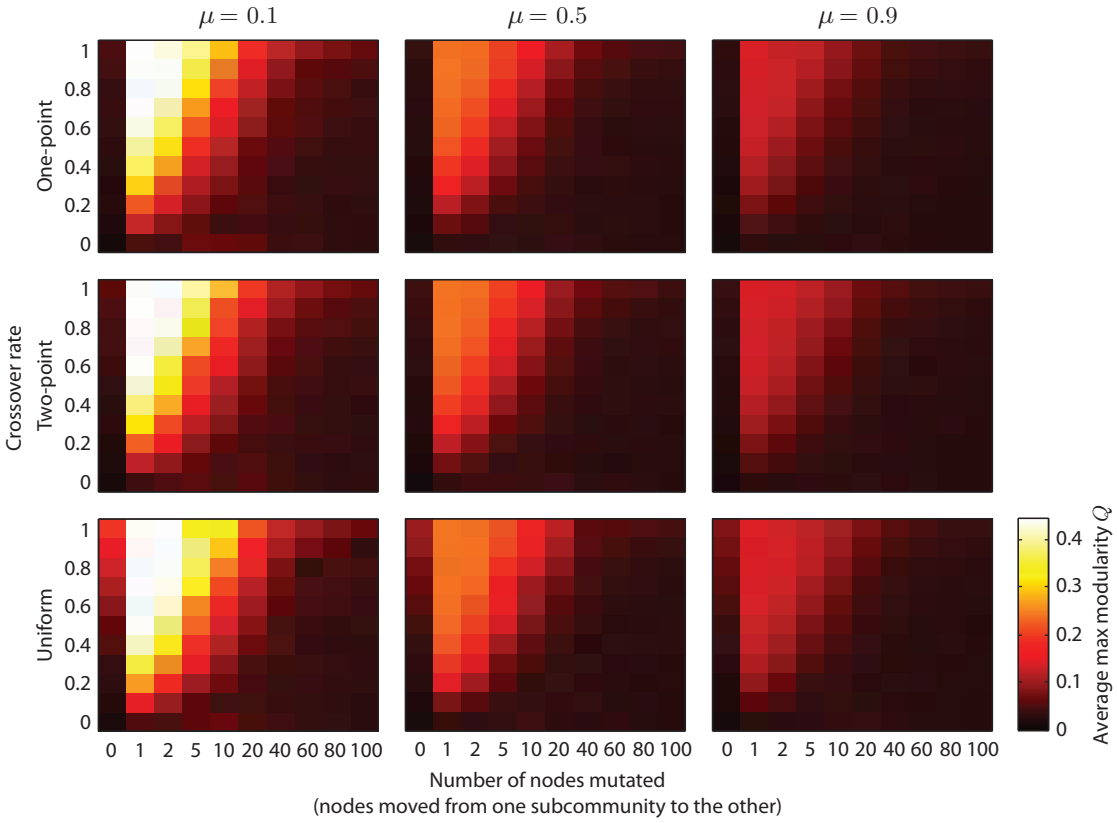


Figure 3.6: Effects of the crossover and mutation methods on the performance of the GA-based module detection method. We apply the GA-based module detection method described in **Section 3.2.2** to identify the optimal splits of 1000-node LFR graphs with high, medium, and low degrees of modularity, which is inversely proportional to the value of the mixing parameter μ. Here we evaluate the performance of one-point, two-point, and uniform crossover with rate in $[0,1]$ with step size 0.1. The rate of the bit-flip mutation operator is defined as the number of nodes mutated in a genome of length c where c the size of the community to split. For each set of experimental condition, we report the average best fitness value obtained over twenty repetitions. For now, the GA stops when the maximum number of generations has been reached which is set to 3000 generations. As expected, we observe that the modularity $Q(\mathbf{s})$ obtained for splitting graphs with high degree of modularity ($\mu = 0.1$) is larger than in graphs where communities are more mixed ($\mu = 0.5$ or 0.9). Moreover, the figure shows that both crossover and mutation operators are required to find optimal split of the network in two subnetworks.

convergence.

When considering a mutation rate set to one node per genome, no significant difference is observed between one-point, two-point and uniform crossover for crossover rates taking values in the range $[0.8,1]$ (Mann-Whitney U-test, $p > 0.05$ for $\mu = 0.1$, 0.5, and 0.9). Therefore, we select the uniform crossover method with crossover rate set to 1 in order to achieve

convergence within the minimum number of generations (**Fig. 3.7**). These crossover method and rate are used in further experiments with the mutation rate set to one bit or node per genome.

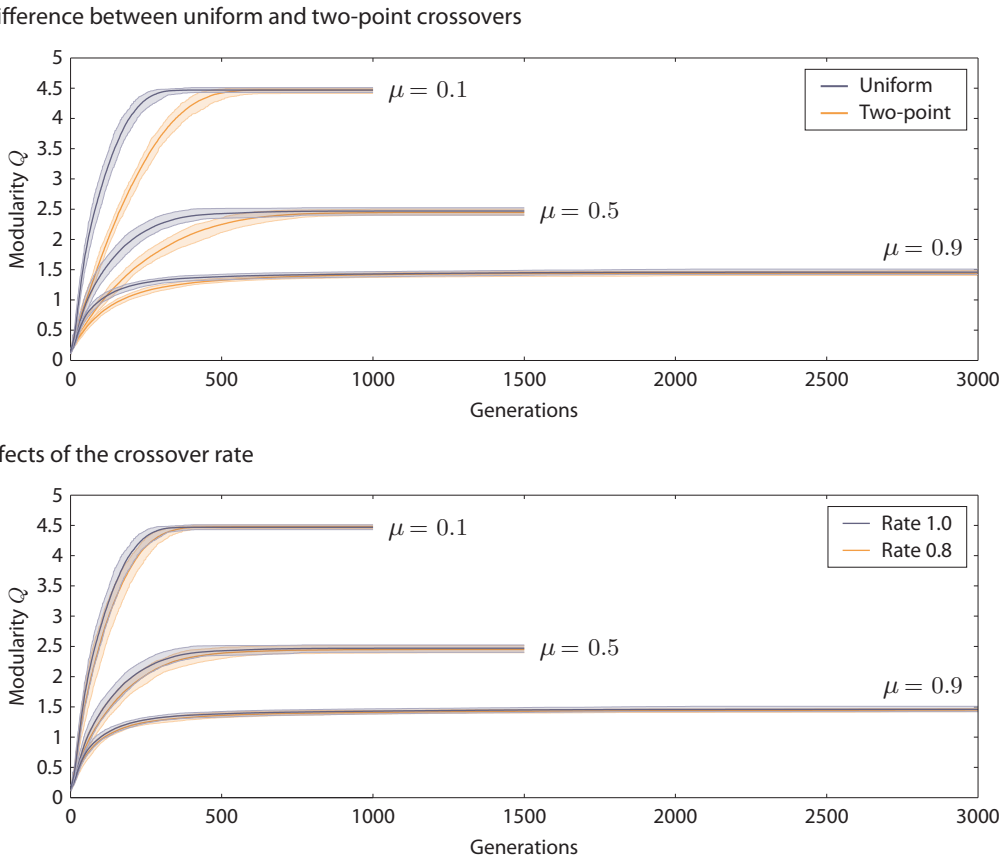


Figure 3.7: Effects of the GA crossover method and rate on the speed of convergence of the GA-based network module detection method. We apply the GA-based community structure detection method described in **Section 3.2.2** to identify the optimal splits of 1000-node LFR graphs high, medium, and low degrees of modularity, which is inversely proportional to the value of the mixing parameter μ. The main lines show the average maximum modularity Q achieved by the population at each generation over twenty repetitions. Moreover, the transparent patch represents the 95% confidence interval (CI). **(A)** We observe that the speed of convergence of the GA is higher when using uniform rather than two-point crossover (crossover rate is set to 1). **(B)** Lower values of the crossover rate reduce the speed of convergence of the GA. Furthermore, no statistical difference is observed between one-point, two-point and uniform crossover for crossover rates set to values in $[0.8, 1]$ (Mann-Whitney U-test, $p > 0.05$ for $\mu = 0.1$, 0.5, and 0.9).

Stopping criteria

There are several criteria that can be used to stop a GA. So far, we set a maximum number of generations that the GA must complete before it returns the best solution found. Other approaches which are often considered consist in stopping the GA when a satisfying solution has been identified or when the convergence has been detected[81,85].

For example, a satisfying solution can often be defined in engineering problems. However, the goal of community structure detection methods is to identify the best partition of a network into modules. More precisely, the approach we adopted consists in finding the partition of a network that maximizes the modularity Q[27], which makes the definition of a satisfying solution to stop the GA inapplicable unless the target partition of the network is known, which is not the case in real applications.

The first stopping criterion used is often to simply wait for the GA to complete a given number of generations or number of individuals evaluated. The implementation of this criterion is straightforward but it comes with several drawbacks. The first difficulty is to define a suitable value for the maximum number of generations. If this number is too small, the GA will not have the time to converge and will return a suboptimal solution. On the other hand, an excessively high value increases the chance of the GA to converge at the expense of wasting computational time. That said, the evaluation of a suitable maximum number of generations also depends on the problem to solve. The higher the dimension or complexity of the problem to solve is, the more generations will be required to find an optimal solution. The design of the fitness function, which provides a quantitative description of the solution space, the choice of the genetic encoding, selection method and genetic operators also affect the number of generations required by the GA to converge. Thus, the identification of a suitable value for this parameter requires to perform preliminary experiments, which must be repeated each time one of the GA parameter changes.

Yet another drawback of this stopping criterion is that the modular decomposition of a network using bi-partitioning methods requires to run the GA each time a community is split in two. If the dimension of the search space for the first split of the network is given by its size n, subsequent module splits typically require fewer and fewer generations as their sizes decrease. As an example, we set the maximum number of generations to 3000 generations in previous experiments to ensure the convergence of the GA for the first split of the networks in two (**Fig. 3.8A**). Eventually the GA is applied to split small modules (10-20 nodes) for which running the GA for 3000 generations does not make much sense. One could think to proportionally decrease the maximum number of generations according to the sizes of the modules being split. However, the evaluation of the maximum number of generation for the first split would still be required, computational time would still be wasted, and the convergence of the GA would never be guaranteed.

Note that the above method is likely to find the optimal decomposition of a network into

modules after an excessively large number of generations. To improve the current efficiency of our GA-based module detection method, we propose to detect when the convergence of the GA occurs, thus allowing to stop the optimization process when improvement of the best solution found is no longer expected. At each generation, we evaluate the diversity of the GA population using the pairwise Hamming distance[94,95]

$$D_h(P) = \sum_{j=1}^{N-1} \sum_{k=j+1}^{N} d_h(i_j, i_k) \tag{3.10}$$

where $d_h(i_j, i_k)$ is the Hamming distance between the individual genomes. $D_h(P)$ is defined for a population of size N with genomes of fixed length l. We then multiply $D_h(P)$ by $\frac{N(N-1)}{2}$ to compute the average Hamming distance between two individuals. Since we use a binary encoding, the average Hamming distance is the average number of bits that differ between two genomes and so the average number of nodes that are distributed differently between two partitions of a community in two subcommunities. Therefore, both genetic and phenotypic diversities are equal here.

The average Hamming distance is reported in **Figure 3.8B** for the modular decomposition of three 1000-node LFR benchmark networks produced for different values of the mixing parameter μ. Initially, the GA population is composed of random individuals and so the expected Hamming distance between two individuals is 500 bits. We observe that the diversity of the population then rapidly decreases as the GA converges. The inset in **Figure 3.8B** provides a four times magnification whose bottom left corner matches the origin of the entire plot. We observe that it takes less generations for the GA to partition modular networks than networks where communities are more mixed ($\mu = 0.5$ and 0.9).

We want the GA to stop automatically when the diversity goes below a given threshold. After having observed the evolution of the diversity for networks with different modularity values (**Fig. 3.8B**), we decide to make the GA stop when the difference between two individuals is on average less than one bit. At that stage, we do not expect the solutions found by the GA to significantly improve because the amplitude of the mutation rate is here relatively small (one node mutated per genome) at least when splitting large communities. By comparing the modularity Q of the split vectors **s** found by the GA using the new stopping criterion with the modularity of the split vector **s** found at generations 1000, 1500 and 3000 depending on the respective μ value of the benchmark networks considered, we conclude that there is no statistical difference between them (Mann-Whitney U-test, $p > 0.05$ for $\mu = 0.1$, 0.5, and 0.9).

Figure 3.8C shows at which generations the GA reached the convergence and stopped. This figure also shows the amount of computational time saved compared to when the GA was undergoing a fixed number of generations. It is interesting to note that the convergence is achieved even faster when bi-partitioning smaller communities, which reduces further the computational time required to partition the network. Finally, the detection of communities

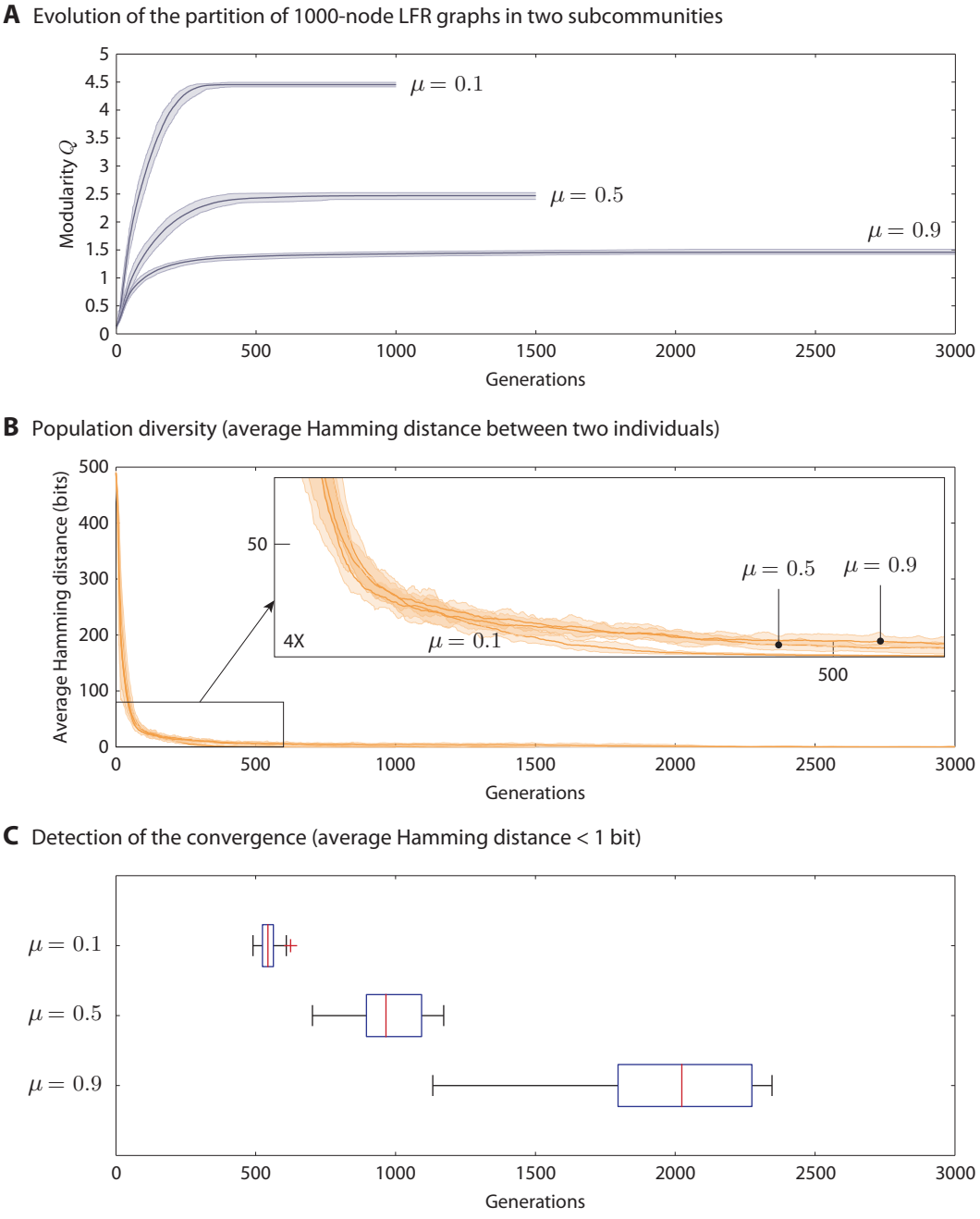


Figure 3.8: Illustration of the diversity-based GA stopping criterion. **(A)** Evolution of the modularity Q for the best split vectors $\mathbf{s}$ found by the GA for networks with modularity values. Here the GA stops after a fixed number of generations. **(B)** We compute the average Hamming distance as a measurement of the population diversity. **(C)** The GA stops by itself when the average Hamming distance is less than one bit, i.e. when the difference between two individuals is on average lower than one bit. The main lines represent the average maximum modularity Q found over twenty repetitions of the GA along with the 95% CI.

in modular networks is more clearly defined than in networks where modules are more mixed (μ = 0.5 and 0.9), which results in an increased speed of convergence towards more clearly identified optima. In networks with lower modularity value, the bi-partitioning task to solve is more ambiguous because there exist many split vectors **s** that achieve very similar fitness values $Q(\mathbf{s})$.

Strategies for preventing premature convergence to local optima

Here we propose two strategies to prevent premature convergences and improve the ability of the GA to find the optimal partition of the network into modules.

As mentioned previously, there is always a risk that the GA will prematurely converge towards a local optimum instead of the global optimum. Therefore, we perform several independent GA runs for each community to split in two communities. We then select the partition that has the largest modularity value Q or generalized modularity value ΔQ.

The second strategy consists in applying the brute force method introduced in **Section 3.2.3** to split small communities. The maximum number of individuals evaluated by the GA during one run is given by the product of the maximum number of generations and the population size. Thus, the brute force method is applied instead of the GA each time the following condition

$$2^{c-1} \leq \text{Maximum number of generations} \cdot \text{Population size} \tag{3.11}$$

is satisfied, where 2^{c-1} is the total number of split vectors **s** to evaluate using the brute force method and c is the size of the community to split in two. Because the number of candidate solutions to evaluate doubles each time the size of the network is larger by one node, the application of the brute force method is limited to relatively small communities, typically smaller than twenty nodes for the identified GA parameters which are summarized in **Appendix C.1.4**.

3.4.4 Evaluation of the refinement techniques MVM and gMVM

We perform a first experiment using small social networks to evaluate the respective performance of Newman's spectral algorithm and the refinement methods MVM and gMVM. Moreover, the brute force method described in **Section 3.2.3** is applied to find the global optimum of each community split in two subcommunities.

Figure 3.9 shows the structure of small social networks, whose size remains limited to enable the detection of their community structure using the brute force approach within a reasonable time (**Appendix C.4**). The three social networks describes friendship relations between different people. The first network has been built to account for the development and resolution of

a conflict among men working on the surface in a mining operation in Zambia[96]. The conflict centered around two persons, P1 and P2. Most of the other workers ended up supporting P1. This can be explain by the fact that P1 had good relationship with four *influent* workers who had themselves good relationship with at least three other workers (**Fig. 3.9A**).

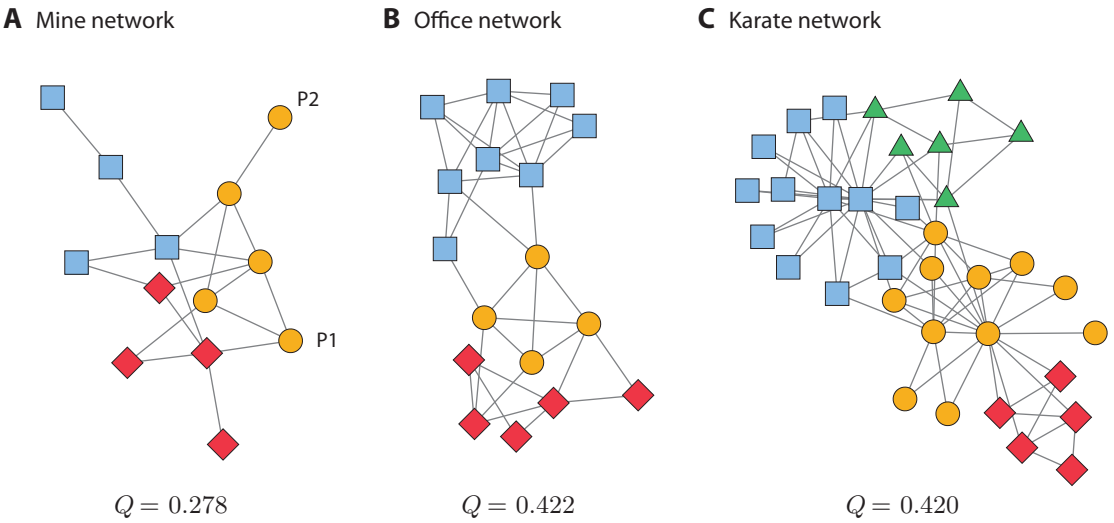


Figure 3.9: Community structure detection in social networks using an improved version of Newman's algorithm. **(A)** Data collected in 1969 to account for the development and resolution of a conflict among men working on the surface in a mining operation in Zambia[96]. The conflict centered on two persons, P1 and P2. **(B)** Cognitive social structure of 21 management personnel in a company to evaluate the effects of a recent management intervention program (edges show friendship relations)[23]. **(C)** Identification of factions emerging after an argument occurred between the members of a karate club[79].

The second network represent data collected from twenty-one managers working in a company manufacturing high-tech equipment[23]. Different questions were asked to the managers to assess the effects of a recent management intervention program. One of them was "Who is your friend?" and the answers collected were used to build the networks shown in **Figure 3.9B**. The modules identified shows that managers belonging to the same department share more friendship relation with other managers of the same department than with others[23].

Figure 3.9C displays the structure of Zachary's karate club network[79], which has been previously used to illustrate community structure inference using Newman's spectral algorithm (**Section 3.2.1**). At some point, an argument occurred between the 34 members of a karate club in the US, which then led to the creation of two clubs, each of them taking almost half of the members. The inferred community structure of this network actually matches accurately the two main factions (blue+green and yellow+red) that have actually emerged after an argument between the members[79].

We report in **Table 3.1** the modularity Q computed for these three social networks using

different methods. First, we show the performance of Newman's spectral algorithm without refinement, then improved using only MVM, and finally we show the performance of the improved version of Newman's algorithm (i.e. *Spectral + MVM + gMVM*). By selecting **s** to be parallel to the leading eigenvector v_1 of the modularity matrix **B**, the Newman method finds only an approximation of the best partition of the mine and karate networks. When selecting MVM, each split of communities in two subcommunities is refined and the quality of the community structure inference matches the quality of the brute force method.

Table 3.1: Community structure detection in social networks. The refinement techniques MVM and gMVM are successively applied to improve the performance of Newman's spectral algorithm. The brute force method is then applied followed by the refinement method gMVM. The networks show the friendship relations between men working on the surface in a mining operation in Zambia[96], management personnel in a high-tech company[23], and the members of a karate club in the US[79].

			Modularity Q				
Network	Nodes	Edges	Spectral	+ MVM	+ gMVM	Bf	+ gMVM
Mine	15	38	0.253	0.256	0.278	0.256	0.278
Office	21	60	0.422	0.422	0.422	0.422	0.422
Karate	34	78	0.393	0.419	0.420	0.419	0.420

Finally, we apply the optimization technique gMVM to further refine the quality of the module detection performed using *Spectral+MVM* and the brute force approach. We show here this last refinement stage still improves the performance of the community structure detection. This also happens to be the case when gMVM follows the application of the brute force method. Therefore, we demonstrate that perfectly splitting communities in two in the sense of maximizing the modularity Q does not lead necessarily to the best partition of the network. Moreover, we show that the refinement method gMVM can still further improve the partitions of networks that are obtained after perfectly splitting communities in two.

3.4.5 Evaluation on LFR benchmark graphs

Here we profile the performance of the community structure detection methods implemented in Jmod on LFR benchmark graphs that we have generated in **Section 3.4.1**. In these networks, the identity of the communities to infer is known.

Figure 3.10 reports the performance of Newman's spectral algorithm refined successively using the refinement methods MVM and gMVM, and the performance of our GA-based module detection method. The x-axis represents the mixing parameter μ and the y-axis the normalized mutual information $NMI(A,B)$ where A is the partition inferred and B is the effective partition of the network (the so-called *ground truth*). The perfect score of 1 is achieved when the partitions A and B are identical (**Section 3.4.2**). Moreover, each line shows

the performance of the methods on one of the four types of benchmarks obtained for different sizes of networks and communities. Each point represents the average normalized mutual information computed from the community structure detection of twenty networks.

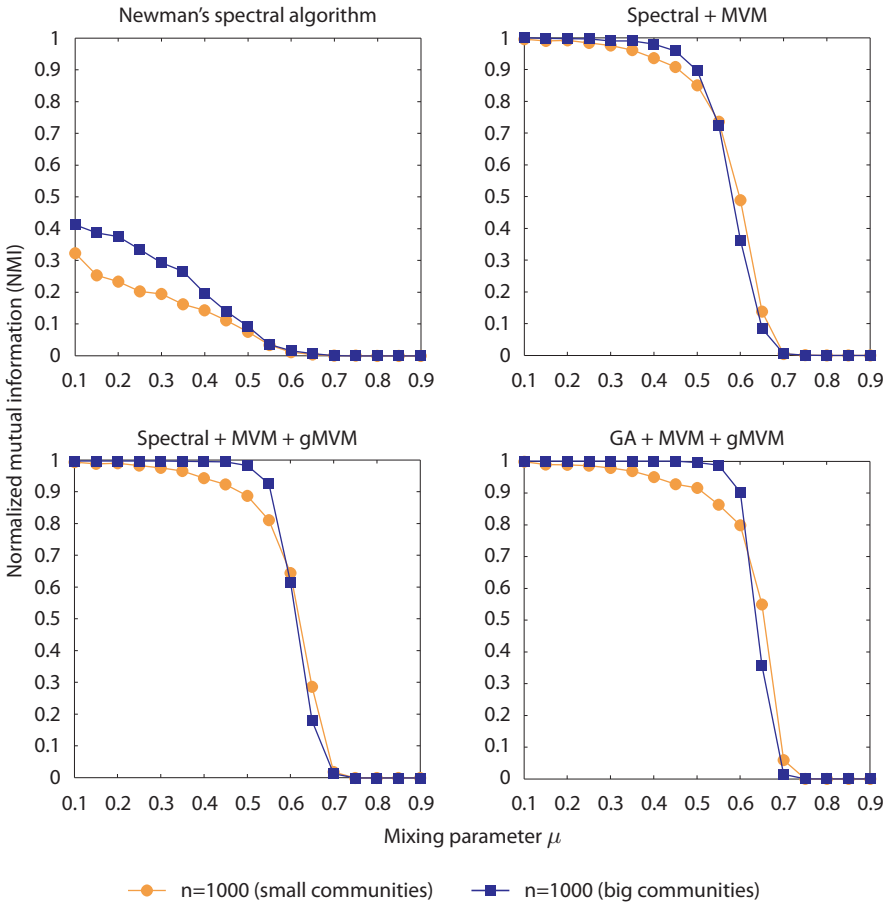


Figure 3.10: Performance profiling of module detection methods using LFR graphs. The benchmark introduced in **Section 3.4.1** is used to evaluate the absolute and relative performance of the methods implemented in Jmod (**Sections 3.2** and **3.3**). The modularity of LFR graphs is inversely proportional to the value of the mixing parameter μ. The benchmark consists of 1000-node graphs composed of small (10-50 nodes) and big (20-100 nodes) communities. Each point represents the average normalized mutual information (NMI) obtained for twenty module detections performed on different graphs.

The first observation is that the performance of the module inference methods is the highest on modular graphs ($\mu \leq 0.5$) than on less modular graphs. The number of edges between nodes that belong to distinct communities increases at the same time μ does. This results in communities more mixed and harder to identify. For $\mu = 0.5$, a node shares as many edges with nodes of the same module than with nodes belonging to other modules as defined by (3.4.1). At that stage, we can wonder if communities are actually well defined in the benchmark graphs. Lancichinetti *et al.* mentioned that all the communities of a graph can be assumed to

be safely defined when

$$\mu \leq \frac{N - n_c^{max}}{N} \tag{3.12}$$

where n_c^{max} is the size of the largest community, respectively 50 and 100 nodes in small and big communities[25]. In the limit case where $n = 1000$ and $n_c^{max} = 100$, communities are well defined for $\mu \leq 0.9$ and so all the graphs considered here at the exception of $\mu = 0.9$ satisfy this condition. It is actually important to note that none of the methods reviewed by Lancichinetti *et al.*[25] achieved NMI values larger than 0 for μ larger than 0.75, which tells us that communities are effectively extremely hard to identify past that mu value.

In the top-left panel of **Figure 3.10**, we observe that applying directly the split vector **s** defined by the leading eigenvector of the modularity matrix **B** does not provide usable module predictions. It is important to note that this method has never been intended to be used without refinement[27]. The contribution of MVM to the performance of Newman's algorithm is particularly remarkable. MVM is a greedy algorithm[89] that locally refines the given split of a community in two subcommunities similarly. An inherent limitation of greedy algorithms is that their performance relies heavily on the initial solution and the shape of the search space. In our experiment, we show that the spectral method provides a suitable initial point to achieve most of the time the global solution using MVM. Moreover, the application of gMVM further improves the quality of the network partitions.

The performance of our GA-based community structure detection method with MVM and gMVM enabled is also shown in **Figure 3.10**. The best performance is actually achieved by this method which identifies the true partitions of all LFR graphs with big communities for $\mu \leq 0.5$. In comparison, the improved version of Newman's algorithm starts making mistakes in such graphs for $\mu = 0.4$. This performance can be explained by the ability of the GA to accurately converge towards (global) optimum for each split when the improved version of Newman's algorithm only performs local optimization. Furthermore, the number of nodes moved by MVM is much lower for partitions inferred using the GA than the spectral method, which also demonstrates the ability of the GA to reach optimal solutions.

Jmod can be used to export *snapshots* of the network each time the detection method changes its community structure. These snapshots provide visual information that can help to understand the behavior of community structure detection methods, and thus can provide insight into how to improve them. We use this feature to visualize the first split of a 1000-node network ($\mu = 0.1$, big communities) in two communities identified using the GA-based method. **Figure 3.11** shows different partitions of the same network where nodes are painted depending on which communities they have been found to belong to. Because GAs are stochastic optimization methods, different solutions may always been expected for multiple independent runs. However, the interesting part is that each partition does not achieve the same Q, yet we have shown previously that the community structure of all twenty such networks were

perfectly identified (**Fig. 3.10**). Therefore, a bi-partitioning method does not have necessarily to find systematically the split that maximizes Q as long as the two communities identified are consistent. This is an important result for modularity optimization methods that can always miss the global optimum.

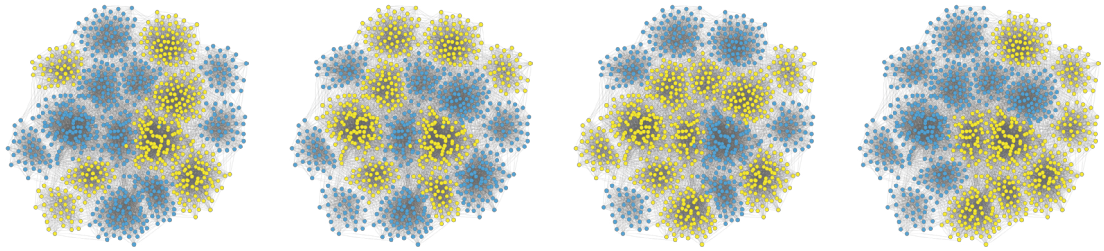


Figure 3.11: First split of the same 1000-node graph in two using our GA-based detection method. We perform four independent runs of the GA to identify the first split of a LFR network ($\mu = 0.1$, big communities) in two. By evaluating the modularity Q of the partitions in (3.1), we conclude that the GA does not always find the split that maximizes Q. However, each of these four GA runs eventually returns the same community structure which matches perfectly the true partition of the graph.

Furthermore, we compare the modularity Q of the true partitions of the graphs with the values found using the GA-based method. The evaluation of Q for the known partition of the LFR graph is achieved through the recursive bi-partitioning of the known communities in two groups similarly to the process performed by Newman's spectral algorithm or our GA-based method (**Fig. 3.12**). As expected, Q is linearly and inversely proportional to the mixing parameter μ.

The second plot in **Figure 3.12** reports the modularity values of the community structures identified using the GA-based method successively refined using MVM and gMVM. The first part matches accurately the values measured for the true partitions of the networks, then values stop to decrease linearly for $\mu \geq 0.6$. This indicates that the method find partitions of the networks into modules that achieves higher Q values than their true partitions. The main reason is that communities are so weakly defined for $\mu > 0.7$ that almost all community structure detection methods known[25] fail to identify them. Past the value of $\mu = 0.7$, LFR graphs become very close to random networks, in which detection methods may still detect a small number of communities due to random fluctuations in their edge distribution.

The issue of finding network partitions that achieve larger modularity Q than the true partitions is also known to be due to a *resolution limit* of modularity optimization methods[77]. In the next sections, we discuss how this limit affects the performance of the above methods and propose a strategy to overcome this limit to some extend using the GA-based module detection method.

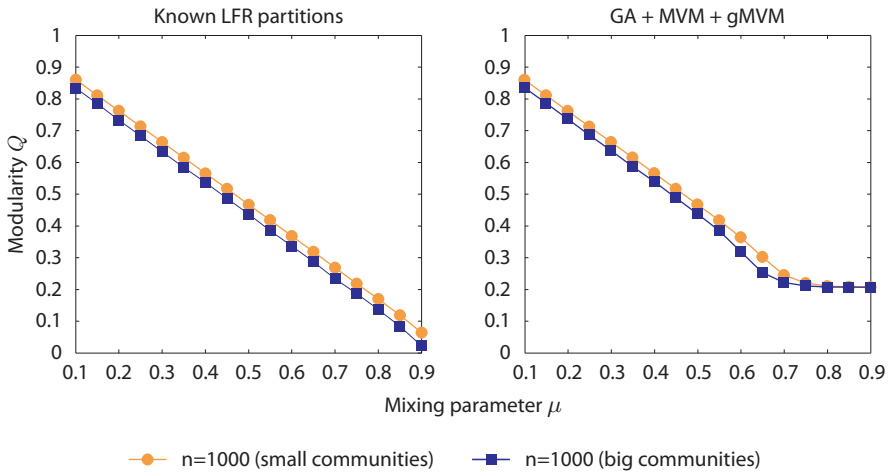


Figure 3.12: Modularity Q of LFR graphs using their true partitions and the GA-based method. As expected, Q is linearly and inversely proportional to the mixing parameter μ. For $\mu > 0.6$, the GA-based method seems to find partitions of the graphs that obtained higher Q values than the true partitions of the networks. Each point represents the average Q value computed for twenty community structure detections on different networks.

3.4.6 Resolution limit of modularity optimization methods

Modularity maximization methods have been shown to be likely affected by a *resolution limit* that makes them fail to identify communities smaller than a given scale proportional to the size of the network and to the degree of interconnectedness of the communities[77,97]. This effect has been shown to occur even for graphs where the natural partitioning of the graph into modules is unambiguous[97].

In the methods introduced in **Section 3.2**, communities are recursively split in two subcommunities according to the split vector **s** that maximizes the the modularity Q for the first split of the network or ΔQ for the subsequent splits. A community is then considered indivisible if splitting it further in two would not contribute positively to the modularity Q. However, Good *et al.*[97] showed using Girvan-Newman's modularity definition[26] that merging two subcommunities into one results in a positive contribution if

$$e_{ij} > \frac{d_i d_j}{2m} \tag{3.13}$$

where e_{ij} is the number of edges between a community i and j and d_i is the total degree of nodes in community i and m is the total number of edges in the network. Thus, considering two communities as one is beneficial from the point of view of modularity maximization if the number of inter-module edges e_{ij} is larger than the quantity $d_i d_j / 2m$ independently of the internal structure of the communities i and j. Evidences of the existence of this resolution

limit are reported by Lancichinetti *et al.*[90] who observed cases where the modularity of the natural partition of a graph is lower than the optimal one found by a modularity optimization method. As a consequence, the number of modules detected is usually smaller than the effective number of communities.

Hence, **Figure 3.13** shows the number of communities in the true partitions of 1000-node LFR networks including small (10-50 nodes) and big (20-100 nodes) communities and the number of indivisible communities detected using the GA-based method. The number of small communities detected is systematically smaller than expected even when modules are clearly defined (e.g. $\mu = 0.1$). This difference increases as networks becomes less modular, which correlates with the degradation in performance of the method observed in **Figure 3.10** (Pearson's correlation, $r = 0.991$, $p < 0.05$). However, the modularity computed by the detection method is not significantly different from the true modularity for $\mu \in [0.1, 0.6]$ (Mann-Whitney U-test, $p > 0.5$). The GA-based method actually finds slightly larger modularity values than the true partition of the network (the different is on average $2.577 \cdot 10^{-4}$). Thus merging two or more communities can have a positive contribution to the modularity Q. This illustrates a serious limitation of modularity optimization methods and show that the quantity Q may not be suitable to infer the community structure of large networks including small communities.

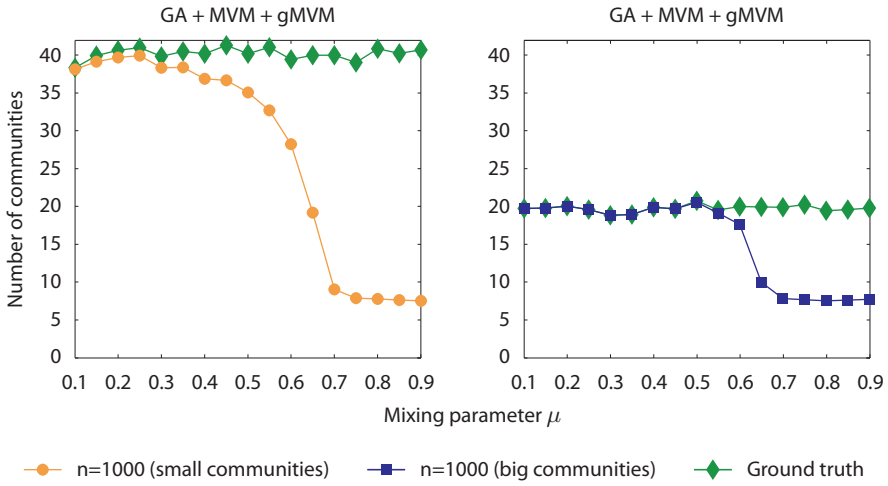


Figure 3.13: Number of modules detected in 1000-node LFR graphs using the GA-based method. The inferred partitions of the networks include systematically less modules than in the true partitions of networks when they are composed of small communities. However, the modularity Q of the inferred partitions is usually slightly higher than the true partitions, which indicates that merging one or more communities can have a positive contribution to Q. For larger modules, the GA-based method always identify the true partitions of the networks for $\mu \in [0.1, 0.5]$. Small and big communities include respectively 10-50 and 20-100 nodes. Each point represents the average number of modules found in twenty different graphs.

For larger communities, **Figure 3.13** shows that the detection method accurately identifies every single module of the networks for μ values in $[0.1, 0.6]$. The number of modules detected

then decreases along with the performance as the degree of interconnectedness of the modules increases. This number reaches a limit for $\mu > 0.7$, which seems to correspond to the number of modules that most of the methods reviewed by Lancichinetti *et al.* achieve on randomized networks[25], this indicating that communities are effectively much harder to detect for $\mu \geq 0.7$.

The above results show that the modularity Q fails to quantify the effective community structure of 1000-node LFR graphs when communities are smaller than twenty nodes (big communities include 20-100 nodes). The degree of interconnectedness of the modules, which increases with μ, also hinders the ability of modularity optimization methods to infer correctly the true partition of the networks. However, previous studies have also shown that similar methods can fail to unravel the effective partition of highly modular networks[77,97]. **Figure 3.14** represents a graph composed of $k = 22$ cliques (cliques are groups of nodes that are fully connected) including each $c = 5$ nodes. The cliques are attached to one another by a single connection to form a ring network[77]. The modularity Q achieved by the natural partition of this graph is 0.864.

We apply Newman's spectral algorithm to identify the community structure of ring network. If the network includes less than twenty-two 5-node cliques, the method successfully infer its true partition where each community are well separated. However, the method starts to make mistakes in larger ring networks (**Fig. 3.14**). In this case, the partition of the graph into modules is not the one where each community consists in a single clique. The modularity Q computed by Newman's algorithm is 0.864, which is $5.551 \cdot 10^{-16}$ higher than the Q value of the natural partition of the graph. This example illustrates well the resolution limit of modularity optimization methods and the inefficiency of the modularity Q to quantify accurately the community structure of all networks.

3.4.7 Community voting method for overcoming the resolution limit

In the previous section, we have discussed how the resolution limit of modularity optimization methods affects the accuracy of network module inference. Here we propose a method to combine multiple partitions of the same network that can overcome to some extend the resolution limit and produce more robust and reliable network module inference.

In **Figure 3.14**, we have shown that Newman's spectral algorithm is unable to accurately predict the community structure of a ring network composed of twenty-two 5-node cliques. The predicted and true partitions of the network into modules are different, yet they both obtain very close Q values. Because Newman's algorithm is deterministic, independent runs would always returned the same incorrect partition. Good *et al.* showed that the modularity function Q of ring networks but also biological networks does not have a peak on top of which the optimal partition is clearly located[97]. Instead, the function Q has a *plateau* where many different partitions have close Q values[97]. Therefore, deterministic methods often get trapped in a suboptimal solutions.

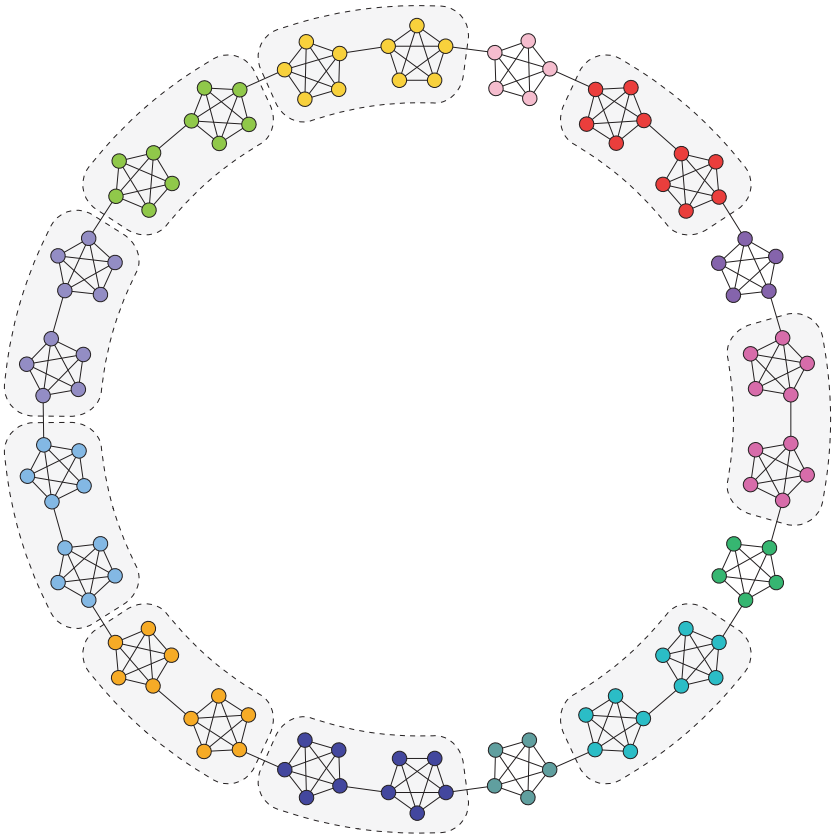


Figure 3.14: Illustration of the effect of the resolution limit using a modular clique ring network. This graph is composed of $k = 22$ cliques including each $c = 5$ nodes. Newman's spectral algorithm is applied to identify the community structure of this graph, however it fails to detect each clique as an individual community (ensemble with gray background are identified as a single community) despite the high modularity value of $Q = 0.864$ of the graph. This is due to its resolution limit that comes from the definition of the modularity Q which fails to quantify accurately the community structure of large graphs including small communities, for instance. It is actually possible to find the true partition of this graph using the GA-based method, however the chance of finding it decreases for ring networks that include more cliques of smaller sizes. It is also interesting to note that the symmetry is here broken, which is most likely due to numerical noise in the computation of Newman's algorithm. A more robust partition inference is obtained by integrating the information of multiple partitions predicted by the GA (**Section 3.4.7**).

The GA-based method that we introduced in **Section 3.2.2** uses stochastic processes to generate candidate partitions of the network. We have shown in **Figure 3.11** that the GA can find different ways to split a network in two subcommunities that obtain similar modularity values. Moreover, the example of the ring network illustrates how the occurrence of the resolution limit can generate different partitions with very close Q values where one or more individual communities are identified as a single community (**Fig. 3.14**). Both observations account well

for the presence of plateaux in the modularity function Q. Therefore, an advantage of the GA-based method over deterministic approaches is that it can be used to sample modularity plateaux. Each point or partition then provides additional information about the community structure of the network.

A challenging task remains the integration of the output of many detections into useful information that would help to unravel the true community structure of the network[98,99]. To address this issue, we propose a voting method to combine N partitions of the same network generated by N independent runs of our GA-based modularity optimization method (**Fig. 3.15**). Here we make the assumption that communities are grouped differently by our method when the resolution limit occurs. First, a list is created to include all the communities identified in the N partitions (**Figs 3.15A** and **3.15B**). Each community then receives a number of votes equal to the number of times it appears in the N partitions (**Fig. 3.15C**). Communities are then sorted first by their number of votes and then by their size so that communities with the largest numbers of votes and smallest sizes are placed at the top of the list. Finally, nodes are removed from communities if they already belong to a community higher in the list (**Figs 3.15D** and **3.15E**).

It has been shown that modularity optimization methods are affected by a resolution limit that makes them fail to identify communities smaller than a given scale proportional to the size of the network and to the degree of interconnectedness of the communities. The reason is that modularity Q accounts for the former but not the latter, with the consequence that small communities may simply not be detected in large networks.

To illustrate the advantage of the voting method, we extend the example of the ring network by generating networks with different different numbers of cliques ($k = 10, 20, ..., 100$) which each includes a different number of nodes ($c = 3, 5, 7, 9$). **Figure 3.16** shows the normalized mutual information obtained by Newman's spectral algorithm on these networks. We can clearly observe when the resolution limit is reached as its occurrence results in a sudden drop in performance. The difficulty in inferring the true partition of the network increases as the number of cliques increases and as their size decreases. We note that the method successfully identifies all cliques as independent communities in the graph that includes twenty 5-node communities, but largely fails when two cliques are added to the network (**Fig. 3.14**).

We apply fifty times the GA-based method to each network to produce as many predictions of its community structure. It is possible that the effective partition of the network is among the predicted ones, however it can not be identified from its modularity Q or its normalized mutual information, which is not available in real applications. The community voting method is then applied to combine the fifty partitions predicted for each network into a single one before evaluating its NMI. **Figure 3.16** that this approach obtains excellent results on the ring network benchmark. It actually enables the perfect inference of each community in each network including 5-, 7-, and 9-node cliques, thus overcoming the resolution limit. However, the resolution limit is not completely overcome for networks composed of 3-node cliques.

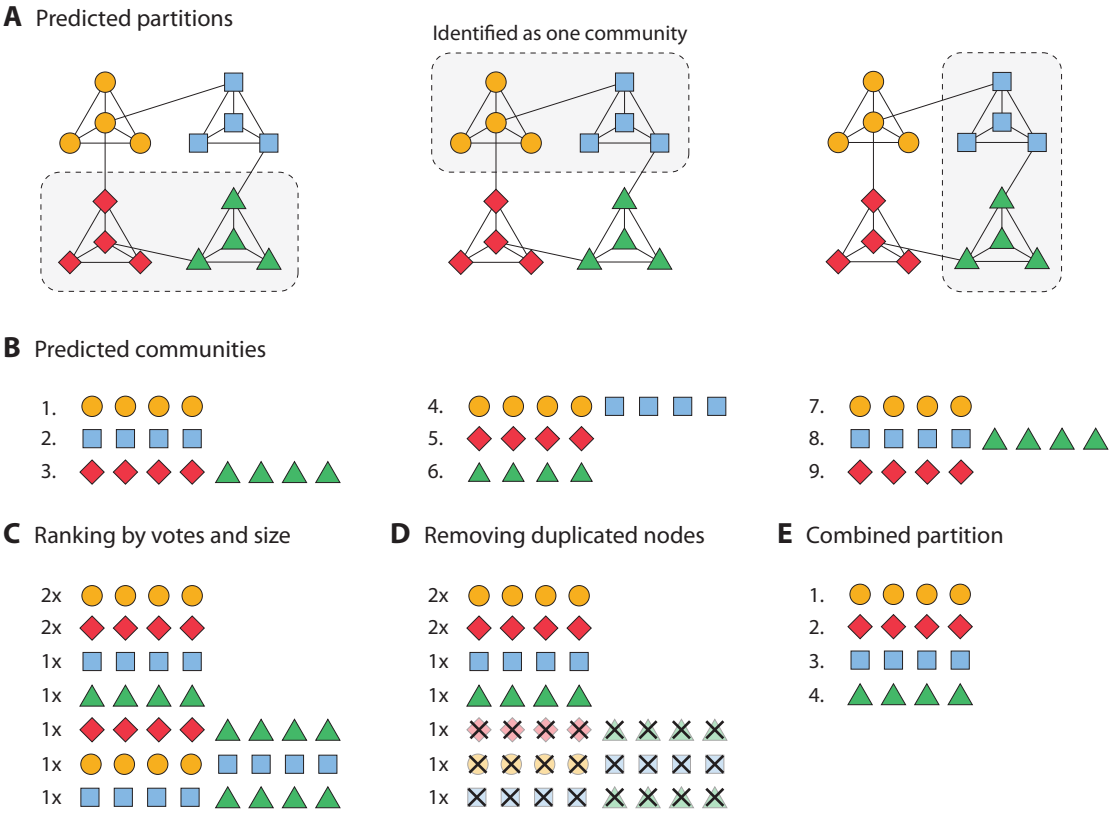


Figure 3.15: Community voting method for overcoming the resolution limit that affect modularity optimization methods. **(A)** The modularity function Q includes a plateau where similar Q values are obtained by partitions where one or more communities may have been mistakenly identified in as one community. We observe that our GA-based community structure detection method has the ability to sample different partitions from the plateau defined by Q. **(B)** List of all the communities that have been identified in multiple partitions. **(C)** Duplicated communities are removed and the remaining ones receive a number of votes equal to the number of times they appeared in the multiple partitions. Moreover, this list in sorted in descending order first by number of votes and then by size. **(D-E)** The combine partition is eventually obtained after removing nodes from communities if they already belong to a community higher in the list.

The explanation lies in the relative frequency of occurrence of each predicted community (i.e. the number of votes). In networks including 5-, 7-, and 9-node cliques, we observed in each predicted partition that the number of cliques successfully identified as communities is on average more than 70%. For 3-node cliques, less than 30% of the cliques are successfully identified so most of the predicted communities are an ensemble of one or more cliques. During the voting process, it is likely than one of this ensemble obtain more votes than a few individual cliques, which then lead to a normalized mutual information lesser than 1. Yet we observed that the accuracy of the integrated partition remains higher than the accuracy of the individual partitions. Moreover, it may be surprising that the accuracy increases with the

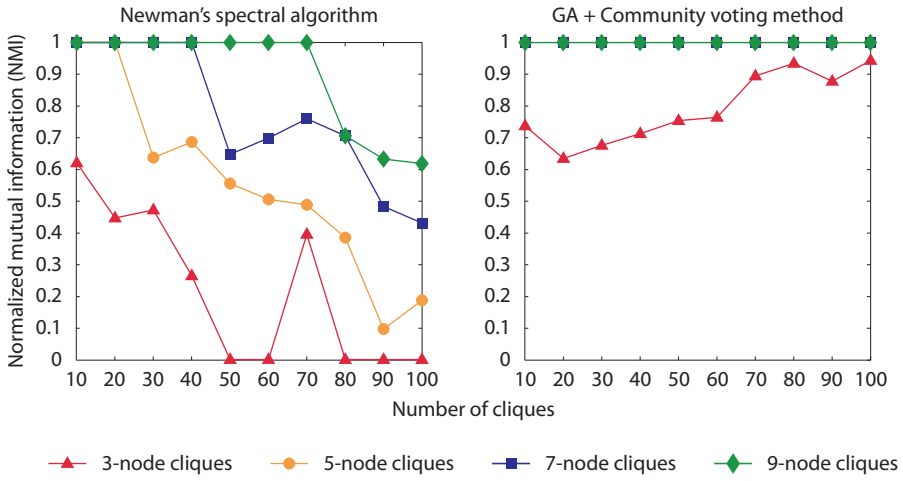


Figure 3.16: Performance of Newman's algorithm and the GA-based method followed by the voting method on ring networks. Networks have been generated to include different numbers of cliques, each of them including a different number of nodes. Newman's algorithm is deterministic, so it returns systematically the same partitions of the network, which is here often incorrect due to the occurrence of the resolution limit. We apply our GA-based method to generate fifty partitions for each network before combining them using the voting method. We observe that our approach perfectly identifies the community structure of graph networks including 5-, 7-, and 9-node cliques, thus successfully overcoming to some extend the resolution limit that affects the GA-based method.

size of the network for 3-node cliques. The reason is that predicted communities that include more than one clique are less likely to be find exactly as they are several times in multiple partitions of larger networks. Such communities would be typically found one or a few more time in the N partitions and receive accordingly a small number votes, placing them behind cliques correctly identified as indivisible communities.

We then evaluate the performance obtained by the GA-based method and the community voting method on the LFR benchmark. The GA-based method is run ten times on each network to predict then partitions, which are refined using MVM and gMVM. The ten partitions are then combine into a single one using the community voting method. **Figure 3.17** compares the performance of the GA-based method alone and when its predictions are used to feed our community voting method. The performance on networks including small communities (10-50 nodes) and big communities (20-100 nodes) are plotted separated to show more clearly the contribution of the voting method.

In **Section 3.4.6**, we have discussed how the resolution limit makes modularity optimization methods fail to identify communities smaller than a given scale proportional to the size of the network and to the degree of interconnectedness of the communities[77]. We have shown that the decrease in performance on networks that include small communities is caused by this

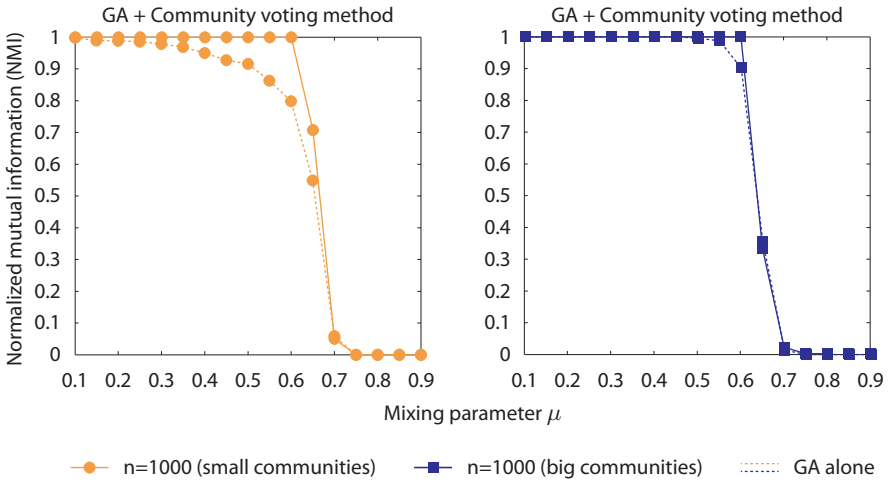


Figure 3.17: Performance improvement of the community voting method on 1000-node LFR graphs. The dashed lines represents the performance of the GA-based method evaluated on the 1000-node LFR benchmark and previously reported in **Figure 3.10**. We have observed that the decrease in performance of the method on networks including small communities is due to the resolution limit which is known to affect all modularity optimization methods[77]. Therefore, we proposed a method for robust and reliable module inference that consists in combining multiple partitions of the same network obtain using our GA-based method by ranking predicted communities according to the number of time they have been detected. The performance improvement is clearly visible especially for networks with small communities where the resolution limit has been successfully overcome.

resolution limit. Even when the GA-based method finds a partition that achieves similar or slightly higher modularity values that the true partition of the network for values of the mixing parameter $\mu < 0.7$, the fraction of communities correctly identified decreases as μ increases (see **Figs 3.12** and **3.13**). The above results demonstrate the application of the community voting method successfully overcome the resolution limit that was affecting our GA-based community structure detection method. Furthermore, the improvement of performance is less important in networks that include big communities because the resolution limit is met.

Finally, the degradation of the performance after $\mu > 0.6$ is mainly due to the fact that community structures are extremely weakly defined. None of the twelve methods reviewed by Lancichinetti *et al.* actually succeeds in capturing these community structures[25]. Moreover, we observe that the performance of those methods for $\mu > 0.7$ are not much different from their performance on random graphs[25]. This is another indication that the community structures of these graphs are either not much different from those of random graphs or the methods applied by Lancichinetti *et al.*, which include state-of-the-art methods[32,35], still lack the sensitivity required to capture such weakly defined community structures.

3.4.8 Module detection in *Drosophila* protein interaction map (DPiM)

Here we apply our GA-based method on the *Drosophila* protein interaction map (DPiM)[21] to detect cluster of proteins. The method identifies fifty-seven modules which are shown in **Figure 3.18**. Nodes are painted in different colors depending on the community they belong to. As a concrete example, all 31 proteins included in the the Snap/SNARE complex manually labelled by Guruharsha *et al.*[21] are correctly identified using our method. Moreover, the detection method suggests that two additional proteins CG7133 (FBgn0037150) and Sgt (FBgn0032640)[100] also participate to the Snap/SNARE complex. We give all the names of the proteins that participate to this complex as as well as their FlyBase[c] identifier in **Appendix C.5**.

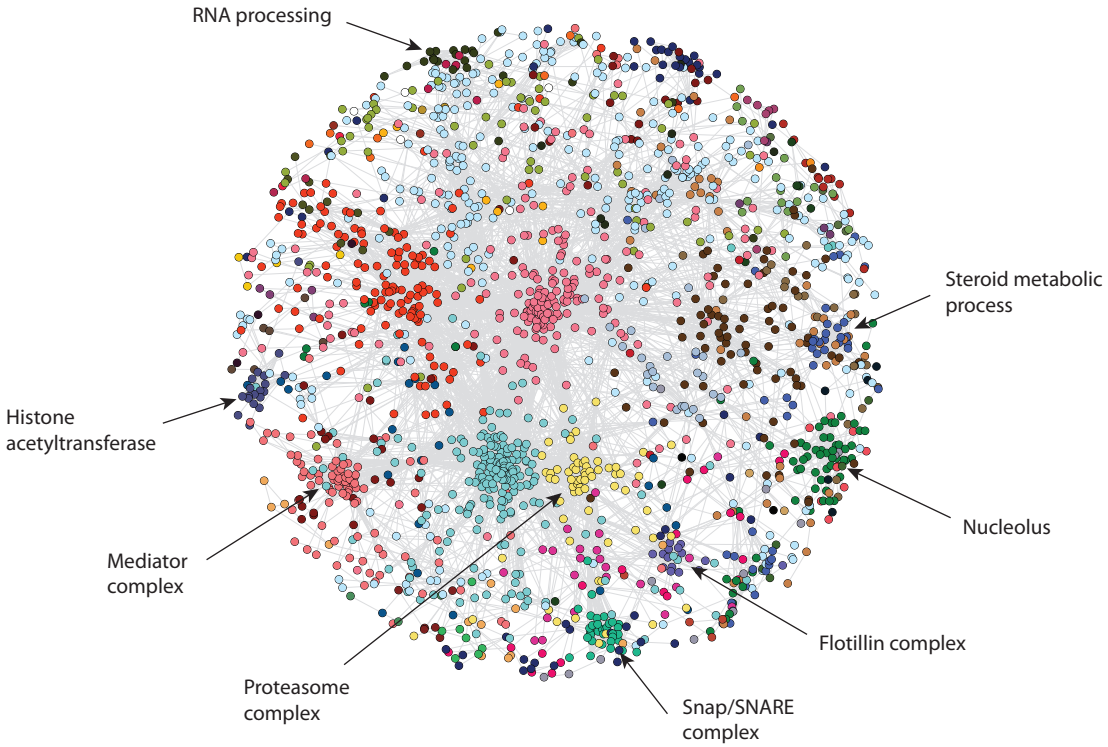


Figure 3.18: Application of the GA-based module detection method to identify cluster of proteins in the *Drosophila* protein interaction map (DPiM). Graphical representation of the largest connected component of the *Drosophila* Protein Interaction Map, which includes 1817 nodes and 10522 interactions[21]. Here we apply the GA-based method to identify clusters of proteins that participate to the same complex. The modularity value computed for this network is $Q = 0.751$. Finally, a deeper understanding of this network could be achieved by comparing the predicted modules to clusters of proteins enriched for GO terms, KEGG pathways or Pfam/InterPro domains[21].

[c] flybase.org

3.5 Conclusions

We introduce an extensible and modular framework for community structure detection in complex networks. We implemented this framework as an open-source Java toolkit called Jmod. Community structure detection is performed by applying not one but several methods. Jmod initially implements Newman's spectral algorithm, a genetic algorithm-based method, and a brute force method. The performance of these methods is further refined using one of the two refinement techniques called moving vertex method (MVM) and global moving vertex method (gMVM). We also developed a voting method to combine multiple network partitions into one reliable and robust partition.

We have profiled the performance of the above methods using real and artificial networks. First, we have evaluated the respective contribution of Newman's spectral algorithm, MVM, and gMVM to the performance of the improved version of Newman's algorithm (*Spectral + MVM + gMVM*). The spectral algorithm finds an approximation of the split of a group of nodes in two which can then be extensively improved using the greedy refinement technique MVM. Using the brute-force method, we have shown that even the partitions found by optimal bi-partition methods can be further improved, for example using our refinement technique gMVM. Moreover, we have profiled the performance of the improved version of Newman's spectral algorithm and our GA-based modularity optimization method on 1000-node Lancichinetti-Fortunato-Radicchi (LFR) graphs generated to obtain different modularity values. Beforehand, we have identified suitable parameters of the GA using LFR graphs. We have shown that both crossover and mutation operators as well as small mutation rate are required to optimally split communities in two. We have also defined a stopping criterion based on the Hamming distance between two genomes. The performance of the GA-based method implemented in Jmod has then been shown to outperform Newman's spectral algorithm including as well as its improved version.

Modularity optimization methods are known to be affected by a resolution limit that makes them fail to identify communities smaller than a given scale proportional to the size of the network and to the degree of interconnectedness of the communities. The reason is that the definition of modularity Q accounts for the former but not the latter. As a consequence, the modularity function Q can include a plateau where several partitions of the network achieve very similar Q values. In addition, the natural partition of the network does not necessarily obtain the highest Q values, which can be associated to other partition where one or more communities are mistakenly identified in as one community. We have illustrated this by applying the improved version of Newman's algorithm and our GA-based method to detect cliques in ring networks. Depending on the number of cliques in the network and their size, both methods can fail to identify each clique as an individual community despite obtaining higher Q values than that of the natural partition of the network. Although several attempts have been proposed to fix the resolution limit of modularity optimization methods[101,102], which represent the largest number of community detection methods, none of them solves it in a satisfactorily manner.

An important difference between Newman's spectral algorithm and our GA-based method is that the former is deterministic and thus always converge to the same partition of the network. However, our GA-based method uses stochastic processes to generate different candidates partitions, each of them providing a piece of information that can be used to identify the true partition of the network into communities. We have taken advantage of this feature to sample the plateau of the modularity function Q resulting from the resolution limit, hence obtaining many partitions of the network. Moreover, an important contribution of this work is the development of a community voting method for combining multiple partitions into one robust and reliable partition. By combining the GA-based method with the voting algorithm, community structures can be accurately reconstructed from partitions that may include mistakes, thus overcoming to some extend the intrinsic resolution limit of modularity maximization. Finally, our *GA+voting method* is best performer along with Infomap[35] in a comparative analysis that profiled the performance of twelve state-of-the-art community structure detection algorithms[25].

4 Towards unsupervised and systematic segmentation of biological systems

The reconstruction of a developmental gene network in its spatial context remains a considerable challenge. One of the reason is that this process requires tremendous amount of spatial and temporal gene expression data, which are usually available in very limited quantities due to the inherent difficulty in measuring gene expression in an entire organism.

Another contribution of this thesis is the development of an image processing method for unsupervised and systematic quantification of the morphology and gene expression of the developing *Drosophila* wing, which is a classical model for studying the genetic control of tissue size, shape and patterning. The method described here has been implemented as an open-source software application called *WingJ*[a]. All that is necessary for our method is a stack of confocal fluorescence images (3D image) of the biological system to quantify. First, a parametric model of the morphology or structure of the *Drosophila* wing is inferred from a fluorescent marker. The segmentation method is based on the design of multiple image processing detection modules, each focusing on the extraction of a specific feature of the wing structure including its orientation. We later extended this approach to the detection of the *Drosophila* embryo. We then use the inferred structure model as a convenient coordinate system for measuring gene and protein expression levels. An important feature of the obtained expression maps is that they can be used to compare domains of expression in differentiated systems, for example to visualize the difference in patterns of gene activity between wild type and mutant wings or in wings imaged at different time points during development. Moreover, a robust, multiscale quantitative description of the developing wing is obtained by combining morphological and gene expression information from multiple wings, completed by the output of an automatic cell nuclei detection method that we have developed. We have used the above method to automatically generate robust quantitative descriptions of wild-type and *pent* deficient *Drosophila* wings imaged at 80, 90, 100, and 110 hours after egg laying. Furthermore, we have shown that these quantitative descriptions can be used to unravel the regulatory interactions of a six-gene wing developmental network.

[a] tschaffter.ch/projects/wingj

4.1 Introduction

Over the last decade, high-throughput assays for mRNA expression have opened the door to the inference of regulatory networks by allowing simultaneous measurements of the expression levels of thousands of genes. Technologies such as spotted microarrays[1] and oligonucleotide chips[2] have enabled genome-wide quantification of differential gene expression profiles and, more recently, short read sequencing technologies such as RNA-seq[3] have provided more precise quantification of mRNA levels. Since then, a plethora of methods have been proposed to reverse reverse genetic networks in single cells[4].

The reconstruction of a developmental gene network in its spatial context remains a considerable challenge. The goal is not only to find a network model that predicts the mRNA and protein concentrations observed as performed in single cells but also to unravel the mechanisms that account for the growth and patterning (cell differentiation) of a multicellular organism or organ. The first challenge is related to the development of models of the organisms at a multiscale systems level as determined by the interaction of processes at the molecular, cellular, and tissue level. Once these models are available, the identification of their parameter values by the reverse engineering process requires tremendous amount of spatial and temporal gene expression data, which are usually available in very limited quantities due to the inherent difficulty in measuring gene expression in an entire organism. Moreover, the generation of reliable and robust models requires the collection of large datasets from many instances of the organism to model, which is time consuming and expensive.

We have used as case study the *Drosophila* wing pouch which later gives rise to the adult wing (see **Section 1.3** for an introduction to the development of the wing). Because the processing of hundreds to thousands wings may be required to generate a reliable and robust quantitative description, we have developed a fully automated detection and segmentation method to generate multiscale quantitative descriptions of multicellular organisms and organs (**Fig. 4.1**) available as a user-friendly and open source Java tool called *WingJ*. This method, which is also the principal result of this chapter, is the fruit of an interdisciplinary project that we have initiated between the Laboratory or Intelligent Systems, the Biomedical Imaging Group at EPFL, and the Affolter Lab at the University of Basel.

First, we infer a parametric model of the morphology or structure of the *Drosophila* wing pouch from a stack of confocal fluorescence images (also called *3D image*). Wg-Ptc antibody labelling is used to visualize the contour of the pouch and its anterior/posterior (A/P) and dorsal/ventral (D/V) boundaries. The orientation of the structure in the space of the image is also recovered using *a priori* information about the morphology of the pouch. In addition to provide a wealth of morphological information, the structure model provides a convenient non-orthogonal coordinate system to quantify gene and protein expression levels. In order to obtained a robust description of the wing, an automatic procedure automatically integrates the morphological and gene expression data collected from individual wings to generate a single and robust quantitative description of the pouch. This description is further extended

with the output of a 3D cell nuclei detection and segmentation algorithm that we developed. Finally, we present the quantitative descriptions that we have generated using WingJ for wild type and *pent*$^{2\text{-}5}$ deficient wings imaged at different time points during development.

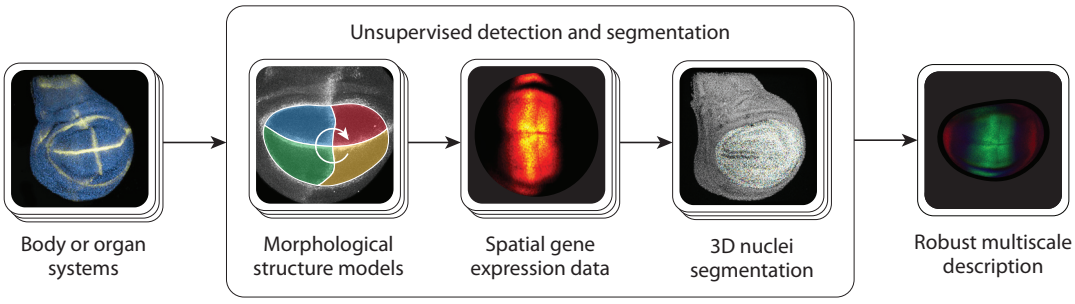


Figure 4.1: Unsupervised segmentation method for generating multiscale quantitative description of biological systems (body systems or organ systems). The method requires a stack of confocal images (also called *3D image*) where fluorescent markers are visible. We first apply a modular segmentation method to infer a model that describes the morphology or structure of the *Drosophila* wing pouch. This model is then used to quantify gene and protein expression. Moreover, the model can be augmented with the output of an automatic cell nuclei detection method that we also developed. The morphological and expression data collected from individual wings are then combined to generate a robust and multiscale quantitative description of the wing pouch.

4.2 Unsupervised segmentation of the *Drosophila* wing pouch

4.2.1 Extensible and modular approach

The method for unsupervised segmentation and detection of the structure of the *Drosophila* wing pouch as delimited by the expression of Wg-Ptc-AB relies on the design of multiple *detection modules*. In this context, each detection module is designed to identify a specific feature from the overall morphological structure to detect (see **Fig. 4.2**). We give here an overview of our approach and discuss the benefits of its modularity.

The first benefit of a modular design is that a detection module can be modified independently of the others as long as it satisfies predefined output specification. A module can be rewritten to implement a different approach expected to improve the reliability of the identification of a target feature. This leads to another advantage which is that the performance of each module can be assessed separately in order to optimize each stage of the overall detection. We evaluated below the performance of two detection modules whose output reliability is crucial for the overall detection on a 50-wing benchmark. Moreover, the image processing tools that we developed (e.g. detection of fluorescent cross-like shapes, detection of a fluorescent trajectory, detection of compartments delimited by fluorescent boundary, etc.) and integrated

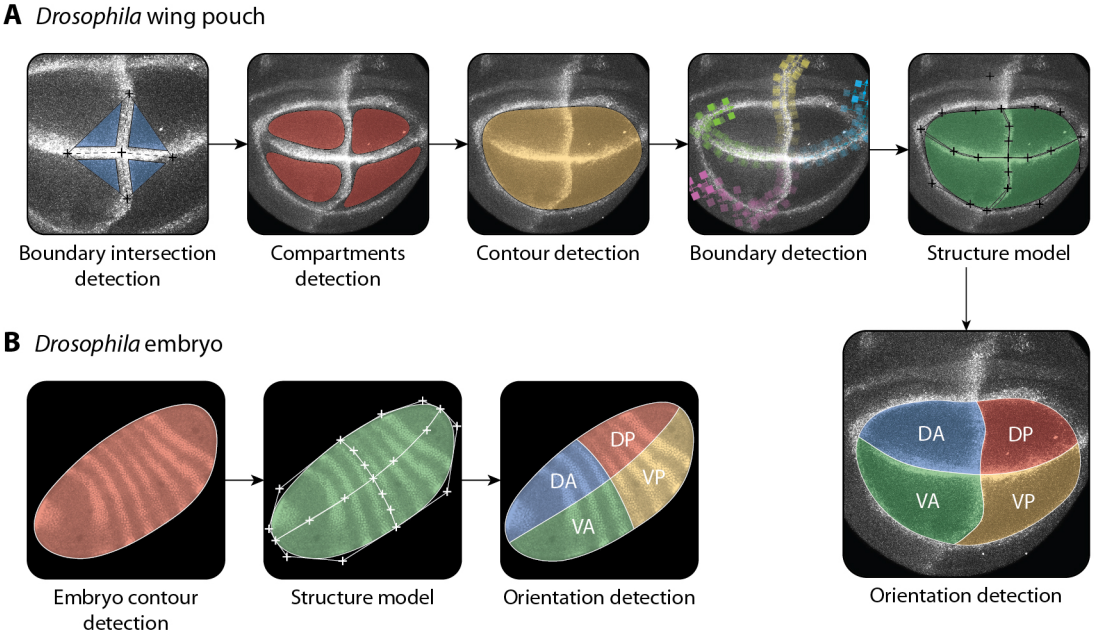


Figure 4.2: Unsupervised detection and segmentation of the *Drosophila* wing pouch and embryo available in WingJ. (A) Detection of the structure of the *Drosophila* wing pouch from Wg-Ptc antibody staining. We designed several *detection modules* to extract different features of the pouch morphology. For example, one module detects the intersection of the A/P and D/V boundaries, one identifies compartments delimited by a fluorescence expression, etc. The output of the modules are then integrated to reconstruct a parametric module of the pouch structure. Generic detection modules can be reused to identify the structure of other systems. **(B)** The detection and segmentation of the *Drosophila* embryo is less complex than that of the pouch and thus can be performed using a smaller number of modules.

into the detection modules are for the most generic enough to target different applications such as the detection and segmentation of other systems.

Furthermore, we observed that our approach achieves high robustness due to partial redundancy between the different detection modules and sharing of information. As an example, the trajectory of the A/P and D/V compartment boundaries inside the wing pouch can be detected by two modules: a first time using the space between the four compartments detected (second image in **Fig. 4.2A**) and a second time by the fluorescent trajectory tracker (fourth image). The issue with the first approach is that spline-based snakes (**Section 4.2.5**) may leak outside of their compartment and explore neighbor compartments when the fluorescence expression along A/P and D/V is too weak. Nevertheless, our method is robust against this type of error as the trajectories of the A/P and D/V boundary can still be recovered using the boundary trackers.

Finally, the above points lead us to suggest that the performance of the detection and seg-

mentation method could be further improved by applying multiple detection modules that aim to identify the same morphological feature (those modules could be applied in parallel to minimize the total computational time). After evaluating their performance on one or several benchmarks, each module can be given a confidence level reflecting the quality of its expected inference. The confidence levels could then be used to select the method to apply, typically the module that achieves the highest performance on the benchmark. An alternative approach would consist in using the confidence levels to weight the output of the different detection modules before integrating them using a consensus or voting method. We have actuall demonstrated in **Chapter 3** that this approach succeeds in performing more reliable community structure detection in networks by integrating the information contained in multiple predicted partitions of the same network.

4.2.2 Preliminary detection

The preliminary detection of the structure of the *Drosophila* wing pouch takes as input a stack of confocal images where the structure has been made visible using one or more florescent markers (**Appendix D.1**). Here, we use Wg-Ptc antibody labelling which is mainly expressed along the A/P and D/V compartment boundaries, and the contour or outer boundary of the pouch.

We denote the stack of confocal images f. The standard coordinate system of an image stack f is defined by the triplet (x, y, z), where x and y are the coordinates of a single image (origin is traditionally located at the upper-left corner of the image) and z describes the depth of the image stack. $f(x, y, z)$ is the value or *brightness* of the pixel located at (x, y, z) within the image stack. From now on, we consider pixels taking grayscale values in $[0, 255]$ (8-bit encoding) as they were directly provided by the confocal microscope (**Appendix D.1.2**).

The task here is to extract a preliminary and almost always incomplete version of the pouch structure from the confocal images. The extracted information will be later used to help identifying more precisely the A/P, D/V, and outer boundary of the pouch. Because we consider the *Drosophila* wing pouch as a 2D system before wing disc eversion (see **Fig. 1.1**), the 3D information contained in the image stack is condensed in order to obtain a 2D representation. This operation enables the later application of existing image processing algorithms optimized for 2D images which currently discard the third spatial dimension. The 2D image is obtained by projecting the image stack f along the z-axis. Formally,

$$\mathrm{MIP}\left(x, y\right) = \max_{z} f\left(x, y, z\right) \tag{4.1}$$

$\mathrm{MIP}(x, y)$ describes the dominant expression features independently of their location within the volume. This method of projection is referred as *maximum intensity projection (MIP)* and is often used as a volume rendering method for 3D data which has been first introduced in the context of nuclear medicine[103]. **Figure 4.3A** shows the MIP image computed for a wild

type wing imaginal discs where the expression of Wg-Ptc-AB is labelled to visualize the pouch structure.

The MIP image is then smoothed using a Gaussian kernel with standard deviation σ_p pixels in order to reduce the intrinsic noise of the acquisition process[104]. This filtering is extremely important since it will ease the forthcoming task of separating the foreground object from the background. The choice of the σ_p value is typically related to the amount of noise present in the image. A large and inappropriate σ_p value would lead to a large noise reduction but also remove useful information. In order to define a suitable value for σ_p, we define

$$\sigma_p = \frac{d}{2\sqrt{2\ln 2}} \tag{4.2}$$

where d is the *full-width-at-half-maximum (FWHM)* of the Gaussian kernel[105]. Here, we set the value of d to the expected thickness in pixels of the compartment boundaries we want to detect. Note that the image scale is usually encoded in the image files generated by the confocal microscope (**Appendix D.1.2**). Our image processing software WingJ automatically extracts this information before displaying it on the main interface.

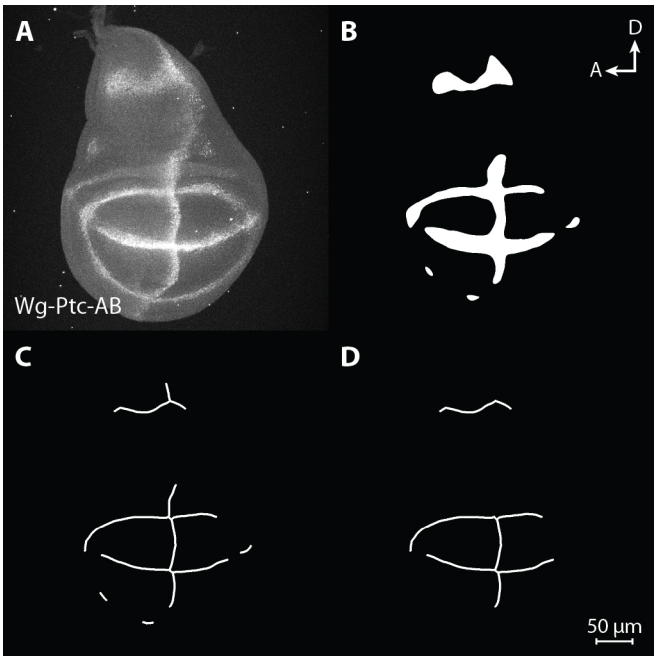


Figure 4.3: Preliminary detection of the *Drosophila* wing pouch structure. **(A)** Maximum intensity projection (MIP) image of a stack of confocal images where the expression of Wg-Ptc-AB is labelled. **(B)** A Gaussian kernel is used to smooth the MIP image before thresholding it through the application of an automatic nonparametric algorithm called *Minimum method*[106]. **(C)** The binary shapes are then skeletonized[107] to extract the relative position of the features while discarding noisy information about their size. **(D)** A pruning algorithm is then applied to remove the smallest filaments of the skeleton.

An automatic thresholding technique is applied to separate the pouch structure from the background. To achieve this, we use a fully automatic nonparametric algorithm called *Minimum method*[106]. The algorithm assumes a bimodal histogram and smooths the histogram iteratively with a running average filter of size three pixels until only two local maxima remain. Histogram classes are denoted by $y_0, y_1, \ldots, y_{255}$ where y_n is the number of pixels in the image whose value is equal to n. The threshold T is the unique minimum value in between two maxima such that

$$y_{T-1} > y_T \leq y_{T+1} \tag{4.3}$$

Figure 4.3B shows the result obtained after thresholding the MIP image using the Minimum method. The white foreground object exhibits a complex shape with uneven boundaries. This comes from the fact that the thresholding is a pixel-wise operation which does not maintain consistency of the foreground as an unique object. This exaplins the presence of artificial holes and discontinuities in the foreground object.

We then use a mathematical structure named *skeleton* to clarify the inherent structure of the pouch. The skeleton structure is a thin version of the thresholded image that is equidistant to its boundaries (**Fig. 4.3C**). The skeleton usually emphasizes geometrical and topological properties of the shape, such as its connectivity, topology, length, direction and width[107–109].

The skeletonization procedure thins a binary image through iteratively removal of pixels[107]. The rules that allow a pixel to be removed ensure that the binary topology is preserved by avoiding the creation of holes and loops. The procedure also takes care of the curve end-points. Indeed, a rule in the removal process ensures that filaments are preserved instead to be eroded to a single pixel. **Figure 4.3C** illustrates the output of the skeletonization algorithm when applied to **Figure 4.3B**.

However, the skeletonization method usually produces many unwanted filaments on the output skeletons that correspond to artifacts generated by the order at which pixels are removed from the structure[110]. Thus, a pruning algorithm is often applied to clean the result of a skeletonization method by removing spurious filaments[110–112]. At that stage of the detection, we want to obtain the skeleton of the A/P and D/V compartment boundaries which will then be used in the next detection module to determine their intersection point. We modified the pruning algorithm described in[110] to introduce a minimum filament length that is used to discriminate between the filaments to be conserved or removed, thus ensuring that smaller filaments are removed while longer ones are conserved. The final output of this detection module is shown in **Figure 4.3D**.

4.2.3 Detecting the A/P and D/V compartment boundary intersection

The aim of this module is to detect the intersection point of the A/P and D/V compartment boundaries, which we refer to as the *wing pouch center*. The present module is applied on the pruned skeleton of the pouch structure shown in **Figure 4.3D**.

Here, we assume that the A/P and D/V boundaries are aligned with the frame of the image as shown in **Figure 4.4B**. This constraint allows to achieve a simple and fast, yet robust detection of the wing pouch center. Actually, it does not matter to the algorithm how the cross-like shape formed by the intersection of the A/P and D/V boundaries is oriented as long as it looks like a straight plus sign '+' on the image. The A/P and D/V orientation of the structure will be inferred at a later stage of the inference method (**Section 4.2.9**). We evaluate below the *success rate* of this detection module using a 50-wing benchmark including wild type and $pent^{2\text{-}5}$ wings imaged at 80 and 110 hours AEL (a sample of this benchmark is given in **Figure 1.2**). We show that the selected approach still achieves a high success rate when a deviation of $\pm 10°$ affects the straight alignment of the '+' fluorescent structure. We also observed that experimenters have typically no difficulties in orienting the wing inside this margin of 20°.

The skeleton of the pouch structure displayed in **Figure 4.3D** is projected on the x- and y-axis of the image (**Fig. 4.4A**). After normalizing independently the two projections, an arbitrary threshold set to 0.75 is defined to detect the dominant *peaks* of each projection. The optimum of each peak is then identified to generate candidate coordinates of the wing pouch center. In **Figure 4.4B**, the projections of the structure generate $N_x = 1$ coordinate on the x-axis (projection of the A/P boundary) and $N_y = 2$ coordinates on the y-axis (projection of the D/V boundary and an additional segment conserved in the skeleton of the pouch structure). The different x and y coordinates are then combined to generate a list including $N_x \cdot N_y$ candidate points for the wing pouch center.

An iterative optimizer is then applied to refine the center candidates before identifying the genuine intersection between the A/P and D/V boundaries. The optimizers are applied in parallel and are initially centered on each center candidates obtained using the above projection method (**Fig. 4.5**). The shape of the optimizers is defined by the expected width of the fluorescent boundaries already specified for the previous detection module. At each iteration, the optimizer moves using a combination of translation and rotation operators to place the Wg-Ptc expression inside the inner plus sign of the optimizer (**Fig. 4.5B**). The convergence is met when the amplitude of the motion of the optimizer falls below a given threshold. The center of the optimizer after convergence then gives the refined candidate point of the wing pouch center.

The second role of the optimizers is to discriminate between the candidate solutions and identify the one that is most likely to correspond to the intersection of the A/P and D/V boundaries. Formally, the ratio $\bar{I}_+ / \bar{I}_{bgd}$ is computed for each center candidate where $\bar{I}_+$ and $\bar{I}_{bgd}$ are the average pixel intensity inside the inner plus sign of the optimizer and its

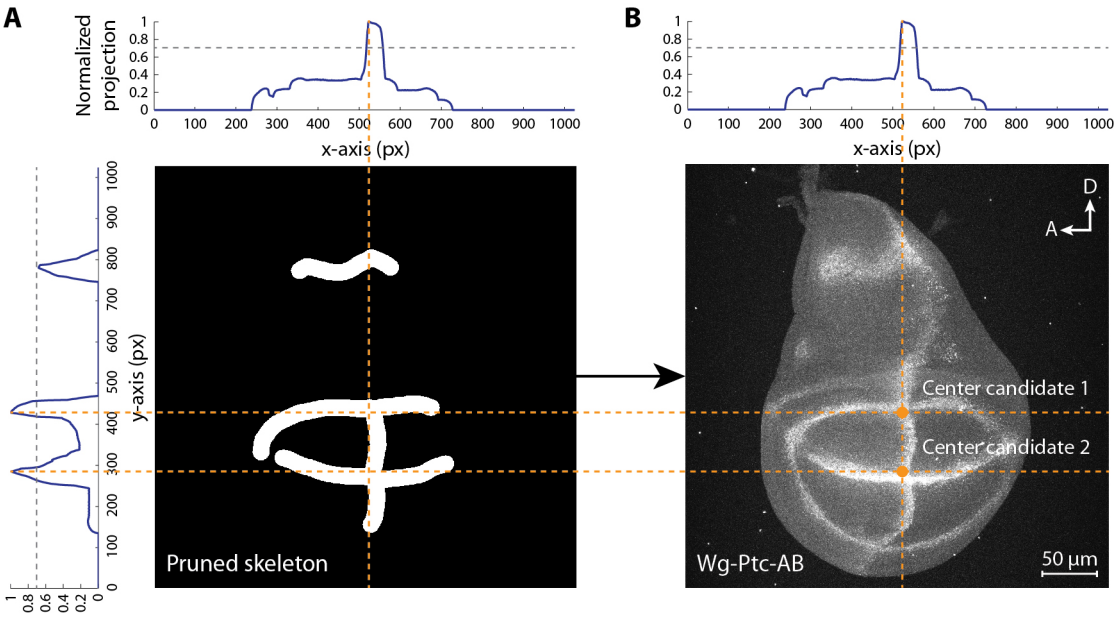


Figure 4.4: Detection of the intersection of the A/P and D/V compartment boundaries (wing pouch center). (A) The wing pouch center detection module extracts the important part of the pouch structure from Wg-Ptc expression while discarding smallest and undesired elements. Assuming that the A/P and D/V boundaries are aligned with the image frame, the skeleton of the wing pouch structure is then projected independently on the x- and y-axis of the image. For each projection, the coordinates of the local optimum of each peaks going above an arbitrary threshold of 0.75 are identified before being combined to generate *candidate points* for the wing pouch center. **(B)** Projecting the skeleton generate here one coordinate along the x-axis (projection of the A/P boundary) and two coordinates along the y-axis (projection of the D/V boundary and an additional segment conserved in the skeleton of the pouch structure). Two candidate points are then generated from the combination of the x and y coordinates. The most plausible point after evaluation (based on a metric) is eventually selected as the effect wing pouch center.

background represented by blue squares (**Fig. 4.5B**). The center candidate with the largest ratio value is then selected and considered as the effective center of the wing pouch structure.

Performance profiling

We evaluate the performance of the wing pouch center detection method using a 50-wing benchmark. The benchmark includes wings imaged at different time points during development (80, 90, 100, and 110 hours AEL) for both wild type (**Fig. 1.2A**) and *pent*$^{2\text{-}5}$ mutant experiments (**Fig. 1.2B**). Note that despite the fact that the *pent*$^{2\text{-}5}$ mutation prevents the normal development of the *Drosophila* wing and leads to smaller wings[50], the mutation does not affect much the structure of the pouch as delimited by the expression of Wg-Ptc. Because the A/P and D/V boundaries are initially aligned with the frame of the image, the benchmark

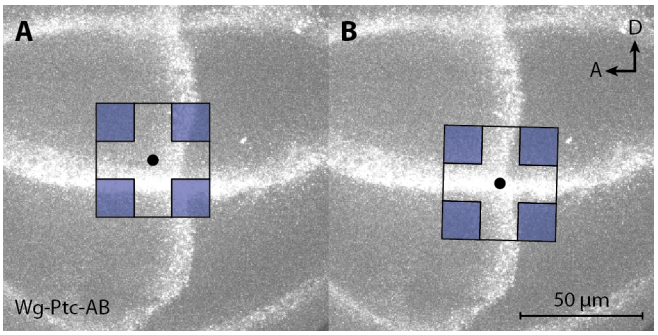


Figure 4.5: Refinement of the pouch center candidates using an iterative optimization algorithm. **(A)** Optimizers are initially centered on each center candidate (**Fig. 4.5B**). The optimizers then start to move using a combination of translation and rotation operators to place the fluorescence expression inside their inner plus sign. The optimization process is deterministic and center candidates are refined in parallel. **(B)** After the convergence of an optimizer, the ratio $\bar{I}_+/\bar{I}_{bgd}$ is computed for each center candidate. Here, $\bar{I}_+$ is the average pixel intensity inside its inner plus sign and $\bar{I}_{bgd}$ is the average pixel intensity falling inside its background area (blue squares). The center candidate with the largest ratio value is finally selected and considered as the effective center of the wing pouch structure.

is extended by rotating each of the fifty wings from $-45°$ to $45°$ with a step size of $1°$. Moreover, another important parameter to evaluate is the expected thickness d in pixels of the fluorescent A/P and D/V boundaries that intersect and form the '+' shape inside the pouch. This parameter as to be specified by the user, therefore we are interested in evaluating its robustness against wild type and $pent^{2\text{-}5}$ mutant wings image at different time points. The range of this parameter is set to [1,50] pixels for testing purpose.

Therefore, our benchmark includes 227'500 images (50 wings × 91 angle values × 50 skeleton thickness values) on which the performance of the detection module is profiled. We manually labelled the intersection of the A/P and D/V boundaries for the fifty wings before applying the present detection module to recover this information from each of the 227'500 *wings* included in the benchmark. A detection is considered successful if the distance between the inferred and true wing pouch center is less than 20 pixels, which corresponds to 7.56 µm in our experiments. The success rate is then defined as the fraction of successful detections over n wings. **Figure 4.6** reports the success rate of the detection module for each experiment.

As expected, we observe that the highest performance is achieved when the A/P and D/V compartment boundaries are aligned with the frame of the image (zero rotation corresponding to the wings as they were imaged by the experimenters). The module achieves a relatively high performance when images are rotated between $-10°$ and $10°$. Larger rotation angles then drastically reduce the performance because projecting the A/P and D/V boundaries on the x- and y-axis of the image does not make sense any more.

We observe that the pouch center is less reliably detected in the youngest wings. This can

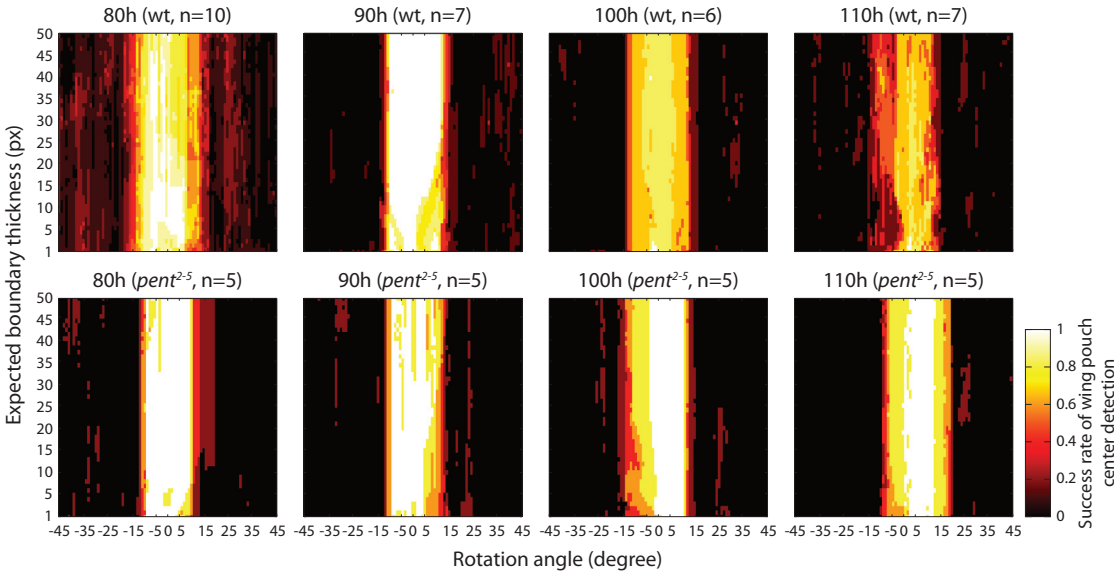


Figure 4.6: Performance profiling of the wing pouch center detection module. The first and second row report the performance for wild type and *pent*$^{2\text{-}5}$ mutant experiments. The columns report the performance for wings imaged at 80, 90, 100, and 110 hours AEL. The success rate of the detection module is reported for each class of experiments and is defined as the fraction of successful detections over n wings. Because we are assuming that the A/P and D/V boundaries are aligned with the frame of the image, we are interested in evaluating the robustness of the algorithm by evaluating the effect of rotating each wing from $-45°$ to $45°$ (x-axis). Zero degree corresponds to the original wings as they were imaged by the experimenters. We also assess the effect of the expected thickness d of the fluorescent boundaries using values from 1 to 50 pixels (y-axis).

be explain by the fact that the structure of these wings is not yet well defined. The pouch center is also more difficult to identify in 110 hours wings. This age actually corresponds to the time of the wing disc eversion, that is, when the 2D pouch system starts to everts in the third dimension, hence nullifying the assumption that the wing pouch can be interpreted as a 2D system (**Video S4**). Moreover, the *pent*$^{2\text{-}5}$ mutation has been shown to inhibit the normal growth of the wings[50]. As a consequence, the structure of the *pent* deficient wings remains more planar than the one of wild type wings of the same age, making the detection module work even more reliably on *pent*$^{2\text{-}5}$ wings than on wild type wings.

To obtain a more general idea of the performance of the proposed detection module, the data shown in **Figure 4.6** are integrated in single plot displayed in **Figure 4.7A**. This panel reports the performance that can be expected from our method independently of the type and age of the wing to process. We observe once more time that despite its relatively simplicity, the approach is robust to the parameters evaluated and achieves high performance ($\geq 90\%$) for wings rotated between $-10°$ and $10°$. **Figure 4.7B** shows the effect of the expected boundary thickness d on the overall performance of the pouch center detection module. A performance above 90% is achieved when setting d to 20 pixels, which corresponds approximatively to

the effective thickness of the Wg-Ptc expression along the A/P and D/V boundaries. More generally, we observe that the expected thickness of the A/P and D/V boundaries is particularly robust, which enables to set it approximatively once and then use it for processing multiple wings. In our experiments, we actually never had to reevaluate this parameter value.

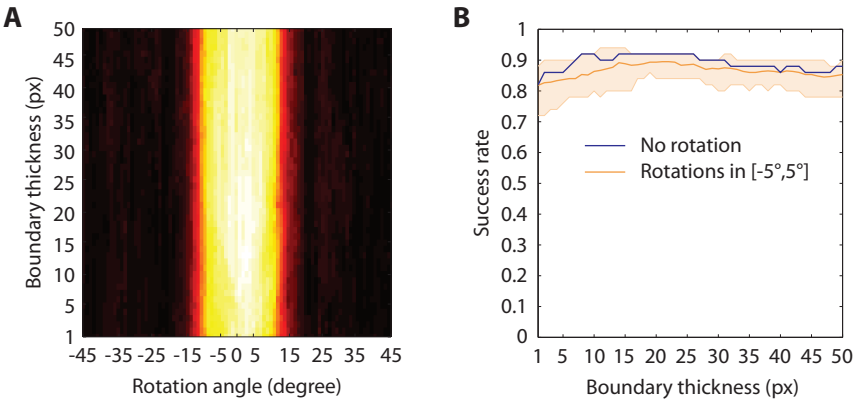


Figure 4.7: Overall performance profiling of the wing pouch center detection module. (A) The overall performance of the detection module is obtained by averaging the eight heat maps from **Figure 4.6**. **(B)** Evaluation of the effect of the expected boundary thickness d introduced in (4.2) on the overall performance of pouch center detection module. The success rate is first reported for the original images only (in blue) and when averaging the results obtained when rotating the original images between $-5°$ and $5°$ (median and 95% CI in orange). We observe that this parameter is quite robust and that a performance above 90% is achieved when setting d to 20 pixels, which corresponds approximatively to the effective thickness of the Wg-Ptc expression along the A/P and D/V boundaries.

4.2.4 Detecting the A/P and D/V compartment boundaries (Part I)

This detection module takes as input the Wg-Ptc MIP image (**Section 4.2.2**) and the intersection of the A/P and D/V boundaries also called wing pouch center (**Section 4.2.3**) to estimate the direction of the four *half-boundaries*. The half-boundaries are defined as line segments that start from the wing pouch center to go towards the dorsal, ventral, anterior and posterior sides of the wing following the trajectories of the A/P and D/V boundaries. The half-boundaries are then used to build a coordinate system within the pouch that will have many applications including the aggregation of many morphological structure models (**Section 4.4.1**) and the systematic quantification of gene expression (**Section 4.3**).

A simple method for separating objects (here the A/P and D/V boundaries) from a background is called intensity thresholding.[113–115] This involves defining one or several threshold parameters whose value can be set manually or derived automatically from the data based on the intensity histogram[113]. This approach can be successful only if the structure to segment and the background are well separated. However, this is not the case in our application since the fluorescence intensity of the A/P and D/V boundaries is diffused. A more elaborate approach

consists in using a predefined *intensity profile*, also referred to as a *template*, to be matched to the image data[116,117]. This method has been shown to work well as long as the shape to segment do not change significantly across different experiments[116]. Here, the shape of the *Drosophila* wing can largely vary due to variation in nutrient quantities ingested during growth, for instance. Also, mutant experiments may sometimes drastically affect the shape of the wing pouch (we have observed that *pent* deficient wings do not grow as much as wild type wings while still conserving a structure similar to the wild type one). A large number of different templates (one for each type of experiments) would be required, thus making the application of the method impractical for the design of a flexible algorithm. Moreover, generating templates would be particularly time consuming and the method computationally expensive[118]. Yet another strategy to segment an image is to apply a *watershed transform*[119]. By considering the image as a topographic relief map and by flooding it from its local minima. This transform subdivides the image into regions (catchment basins) with delimiting contours (watersheds). However, the basic algorithm has several drawbacks such as sensitivity to noise and would have a tendency toward oversegmentation in our application[120]. We actually use a watershed transform for another application of interest: the 3D detection of cell nuclei which we will introduce in **Section 4.5**.

In recent years, there has been an increasing interest in using deformable models or *active contours* in segmentation[121–124]. They can be broadly categorized depending on their shape representation:

- **Parametric active contour models.** Also called *snakes*, their shape is described explicitly by some parameters. These parameters can take different forms depending on the curve parameterization, e.g. control points the snake curve must go through[125–127], Fourier descriptors[128], spline coefficients[129,130], etc.

- **Implicit active contours.** The representation of the curve is implicit and described obtained as the zero level-set $\phi^{-1}(0) = \{(x, y)|\phi(x, y) = 0\}$ of a scalar function ϕ defined over the image domain[131–133].

Snakes are initialized with a generic and/or given shape and iteratively evolve in the image space to optimize a function called *snake energy*. This function usually includes image-dependent and shape-related terms to enable flexible incorporation of image information and prior knowledge. Snakes have been applied to a wide range of segmentation problems[121,122]. The framework is very flexible and can be tailored to each application by designing a specified energy function and incorporating shape constraints in the geometry[134]. Therefore, we decided to rely on snakes due to the fact that they require fewer parameters than the level-set approach and lead to a faster optimization procedure.

Kite snake

We developed a snake parameterized by five control points that exhibits a structure similar to the one of a a kite, so we naturally called it *kite snake* (**Fig. 4.8**). Each branch or segment of the kite snake is designed to capture a half-boundary of the wing pouch, that is, one half of the A/P and D/V boundaries. The control points P, Q, R and S define the outer quadrilateral of the kite snake. The remaining control point C corresponds to the center of the kite snake which should be aligned on the wing pouch center identified by the previous detection module.

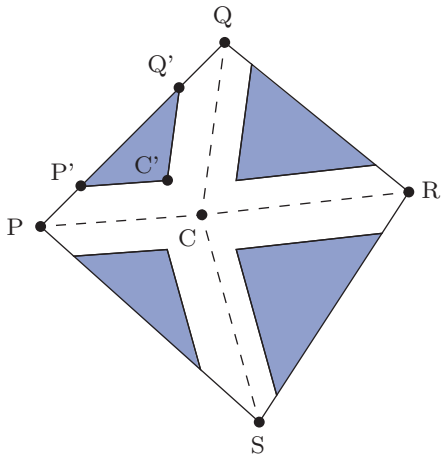


Figure 4.8: Schematic representation of the kite snake model. We developed a segmentation model to identify the direction of each of the four half-boundaries that start from the wing pouch center and go toward the anterior, posterior, dorsal, and ventral sides of the wing. The main feature of the kite snake is that it can be applied to identify cross-like shapes that do not have right angles between them. This is actually the case since the A/P and D/V boundaries are curved and do not intersect perpendicularly. The geometry of the kite snake is determined by the position of its five control points C, P, Q, R, S which are the free parameters of the model. In particular, the shaded triangle $P'Q'C'$ is entirely determined by the outer triangle PQC as defined by (4.5), (4.6), and (4.7).

We consider the four (shaded) outer triangles PQC, QRC, RSC and SPC and the four inner triangles $P'Q'C'$, $Q'R'C'$, $R'S'C'$ and $S'P'C'$ (**Fig. 4.8**). The sides of the outer triangles are constructed by connecting the kite snake points P, Q, R, S and C. The sides of the inner triangles are parallel to their outer counterpart. Namely, the segments CP, PQ and QC are parallel to the segments $C'P'$, $P'Q'$ and $Q'C'$, respectively. Moreover, the area enclosed by each outer triangle and the corresponding inner and outer triangle satisfy

$$\alpha = \frac{\text{Area}\left(P'Q'C'\right)}{\text{Area}\left(PQC\right) - \text{Area}\left(P'Q'C'\right)} \tag{4.4}$$

where $\alpha \in [0, \infty)$ is a parameter that controls the size of the outer region and thus the thickness of the four segments to capture. When $\alpha = 0$, the area of the inner triangle $P'Q'C'$ is zero. On the other extreme when $\alpha = \infty$, the triangles PQC and $P'Q'C'$ coincide. Here, we consider

$\alpha = 1$ which leads the area of the inner triangle P′Q′C′ to be half the one of PQC. In that configuration, the coordinates of the points of the outer triangle are given by

$$\mathrm{C}' = (1-2\delta)\,\mathrm{C} + \delta\,(\mathrm{P}+\mathrm{Q}) \tag{4.5}$$

$$\mathrm{P}' = \mathrm{P} + \delta\,(\mathrm{Q}-\mathrm{P}) \tag{4.6}$$

$$\mathrm{Q}' = \mathrm{Q} + \delta\,(\mathrm{P}-\mathrm{Q}) \tag{4.7}$$

where $\delta = \frac{1}{2}\left(1-\sqrt{1-\alpha}\right)$.

The kite snake is applied to a smoothed version of the Wg-Ptc MIP image and has its center C initially centered with the intersection of the A/P and D/V boundaries previously identified (**Fig. 4.9A**). Once the active contour algorithm has converged, the four segments of the kite snake (white regions in **Fig. 4.8**) should include most of the bright pixels of the A/P and D/V boundaries while conserving approximately their initial size (**Fig. 4.9B**). To achieve this, we define the energy function

$$\begin{aligned} E_{\text{kite}} = & \frac{-1}{\text{Area (Cross)}} \int_{\text{Cross}} \text{MIP}\left(x,y\right) \mathrm{d}x\,\mathrm{d}y \\ & + \left(\|\mathbf{p}-\mathbf{c}\| - l_0\right)^2 \\ & + \left(\|\mathbf{q}-\mathbf{c}\| - l_0\right)^2 \\ & + \left(\|\mathbf{r}-\mathbf{c}\| - l_0\right)^2 \\ & + \left(\|\mathbf{s}-\mathbf{c}\| - l_0\right)^2 \end{aligned} \tag{4.8}$$

where Cross is the region enclosed by the quadrilateral PQRS excluding the outer triangles, and l_0 is the initial length of the branches of the kite snake. The first term in the energy function has a high negative value when the fluorescence of the A/P and D/V boundaries falls inside the inner part of the kite snake. The other terms are quadratic penalties associated to each control points to enforce the conservation of the length of the four segments.

The position of the control points determine the configuration of the kite snake and are tuned using a standard unconstrained optimization algorithm while minimizing E_{kite}[135]. This type of optimization algorithms consider equally all degrees of freedom of the model (here the 2D coordinates of the control points), and perform a line search within the vector space formed by the parameters. In the optimization scheme, a direction within the parameter space is first chosen depending on the partial derivatives of the energy before performing an one-dimensional minimization along the selected direction. Then, another direction is chosen using the partial derivatives of the energy function while enforcing the same conjugation properties. This scheme is repeated until convergence is achieved, i.e. until the variation of the energy function in between consecutive steps of the algorithm falls below a given precision.

The snake is originally centered on the location of the wing pouch center inferred by the

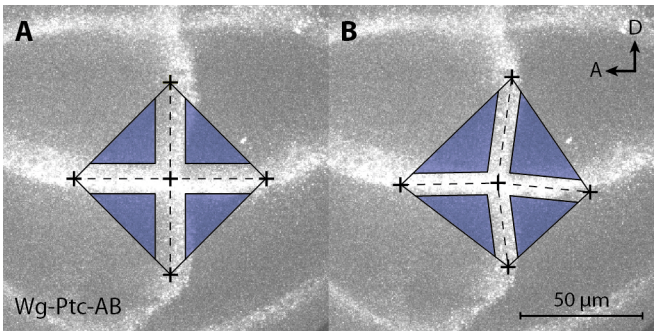


Figure 4.9: Detection of the direction of the A/P and D/V boundaries using the kite snake model. (A) The active contour model we developed and named kite snake is initially centered on the wing pouch center, that is, the intersection point of the A/P and D/V boundaries previously identified (**Section 4.2.3**). The kite snake can be used to find the direction of the four segments of a cross-like shape whose center is approximatively known and even if the angles between any two neighbor segments are not right. **(B)** After optimizing the energy function given by (4.8), the four segments of the kite snake are aligned on the parts of the A/P and D/V boundaries that start from the pouch center. In WingJ, the '+' elements correspond to the control points C, P, Q, R, S of the kite snake model. These points can be manually dragged-and-dropped at any time before and after the optimization of energy function.

previous detection module described in **Section 4.2.3**. The four control points remaining are then distributed in such a way that the initial configuration of the snake matches the geometry of a plus sign (**Fig. 4.9A**). After convergence of the optimization algorithm, the kite snake model provides an accurate description of the intersection of the A/P and D/V boundaries (**Fig. 4.9B**). In our image processing toolkit, the '+' elements correspond to the control points C, P, Q, R, S of the kite snake model. These points can be manually dragged-and-dropped at any time before and after the optimization of energy function if required.

The kite snake model requires the specification of two parameters. The first parameter δ must be set with a value roughly equal to half the expected width of the A/P and D/V boundaries delimited by the expression of Wg-Ptc (δ is set via α). This allows to correctly define the width of the four inner segments of the kite snake. The second parameter l_0 correspond to the initial length of the four inner segments. The next section shows that l_0 is actually robust and can be used across multiple experiments without requiring to modify its value. In all our experiments, we used the value $l_0 = 100$ pixels, which is approximatively equal to the length of the smallest half-boundary observed in the youngest wing discs considered (**Fig. 1.2**).

Performance profiling of the kite snake

The effect of the parameter l_0 is evaluated using the 50-wing benchmark. First, we manually identify the desired configuration of the kite snake model for each of the fifty wings by moving its control points so that the inner region of the kite covers the intersection of the A/P and D/V boundaries (the ground truth). Then, the same detection is performed automatically using

the kite snake optimization technique. We test values of the parameter l_0 between 5 and 400 pixels with 5-pixel increments. **Figure 4.10A** illustrates the selection of $l_0 = 100, 200, 400$ pixels in a 100-hour-old wing. The detection of the A/P and D/V boundaries by the kite snake model is considered successful if the difference between the inferred and true angle between any two neighbor segments of the kite snake is less than 2° (a relatively severe constraint). The success rate of the method is then defined as the fraction of successful detections over the fifty wings. **Figure 4.10B** shows that a minimum length $l_0 \approx 100$ pixels is required to achieve relatively high performance. We observe that the kite snake model is very robust against any value $l_0 \geq 100$ pixels but also for values larger than half the typical length of the A/P and D/V boundaries.

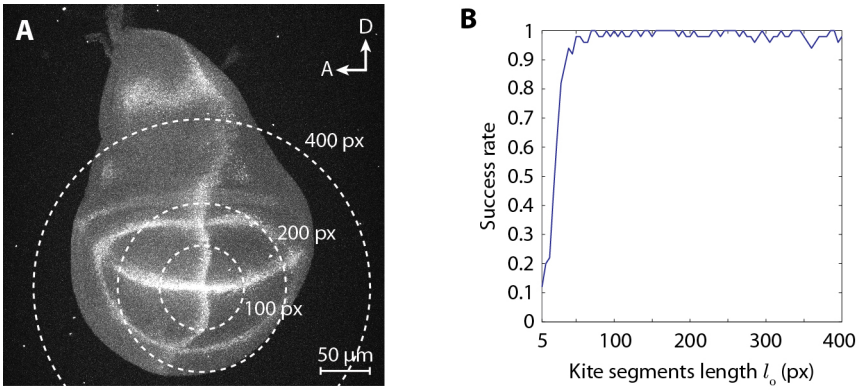


Figure 4.10: Performance profiling of the detection module inferring the direction of the A/P and D/V boundaries. (A) Initially, the kite snake model is centered on the wing pouch center, i.e. the intersection point of the A/P and D/V boundary (**Section 4.2.3**). Three dashed circles are drawn around the wing pouch center with a radius equal to 100, 200, and 400 pixels. **(B)** Here, we observe that a minimum value of 100 pixels for l_0 is required to achieve high success rate ($\geq 95\%$). In all our experiments, we used $l_0 = 100$ without ever reconsidering this value.

4.2.5 Detecting the wing pouch compartments

The goal of this module is to infer a parametric model of the four compartments DA, DP, VA, and VP included in the *Drosophila* wing pouch. In addition to the usual Wg-Ptc input image, this module requires information provided by the kite snake model inferred in **Section 4.2.4**. To be more specific, the kite snake model is used to obtain a parametric description of the cross-like shape formed by the A/P and D/V boundaries in the vicinity of their intersection.

Our approach for identifying the four compartments relies on the development of a second snake model based on B-spline curves[136]. Compared to the kite snake model which has been designed to identify cross-like shapes, the spline-based model allows to approximate closed curve in a plane. One spline-based snake is required to capture each compartment whose contour is delimited by the A/P and D/V boundaries, and the outer boundary of the

pouch, that is, the contour of the pouch as defined by the expression of the protein Wingless (Wg). Thus, four spline-based snakes are required to generate a parametric model of the four compartments. The spline-based snake is initialized with a circle shape and is centered on a point taken between two neighbor segments of the kite snake (**Fig. 4.11A**). Similarly to the kite snake model, an unsupervised optimization algorithm is implemented to iteratively minimize the energy function defined below. **Figure 4.11B** shows an example of the typical configuration of the four spline-based snakes after convergence. In the next paragraphs, we formally describe the optimization technique we developed for unconstraint identification of the wing pouch compartments using spline-based snakes.

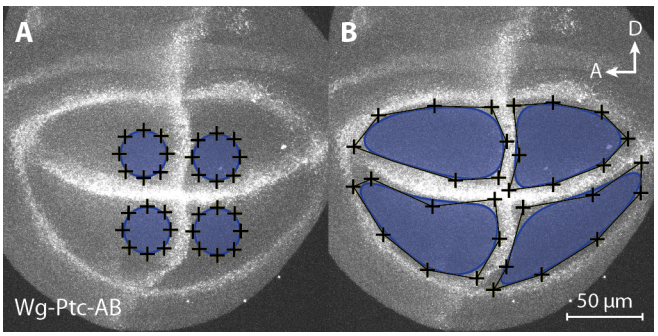


Figure 4.11: Detection of the wing pouch compartments using spline-based snake models. **(A)** The snakes are initialized with a circle shape. The kite snake model described in **Section 4.2.4** provides the information required to ensure that each spline-based snake is initially located inside a different compartment. **(B)** After optimization, the spline-based snakes provide a parametric description of the shape of the four compartments DA, DP, VA, and VP delimited by the fluorescence expression of Wg-Ptc. Here, the effective contour of the snakes is displayed as a solid line enclosing a blue region. The '+' elements represent the control points of the model and their location define the effective contour of the snakes.

Spline-based snakes

A parametric curve in the plane denoted $\mathbf{r}(t)$ is described by a pair of Cartesian coordinate functions $x(t)$ and $y(t)$, where $t \in \mathbb{R}$ is a continuous parameter. We parameterize the one-dimensional functions x and y by linear combinations of basis functions. Among all possible bases, we use those derived from a compactly supported function φ and its integer shifts $\{\varphi(t-k)\}_{k\in\mathbb{Z}}$. These definitions enable the use of fast and stable interpolation algorithms[137].

Considering closed curves, the coordinates functions $x(t)$ and $y(t)$ are periodic, and so is the vector equation $\mathbf{r}(t)$. Then, we build the spline closed curve by specifying an M-periodic sequence of control points $\{\mathbf{c}[k] = (c_x[k], c_y[k])^{\mathrm{T}}\}_{k\in\mathbb{Z}}$, with $\mathbf{c}[k] = \mathbf{c}[k+M]$. The vector parametric representation of the curve is given by

$$\mathbf{r}(t) = \sum_{k=-\infty}^{\infty} \mathbf{c}[k]\,\varphi(M\,t-k) \tag{4.9}$$

where the period has been normalized to the unity. M represents the the number of control points, and determines the degrees of freedom in the model. Small values lead to simpler and constrained shapes while larger values provide more flexibility and accuracy in the model. **Figure 4.12** shows a curve parameterized by a few control points as well as its corresponding coordinate functions.

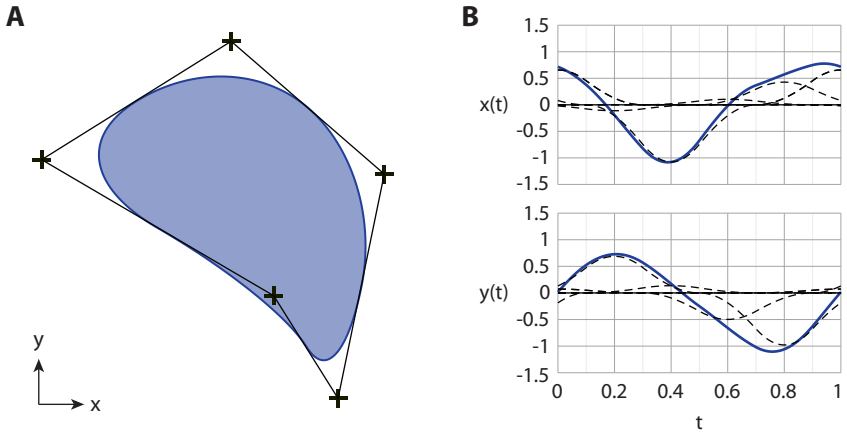


Figure 4.12: Parametric representation of a closed curve in spline-based snake models. (A) The parametric model we developed for describing closed curves is given in (4.9) (here, $M = 5$ control points). The contour of the spline-based snake is shown as a solid line enclosing a blue region. The control points $\{\mathbf{c}[k]\}_{k\in\mathbb{Z}}$ of the snake model are represented by '+' elements. **(B)** The parametric functions $x(t)$ and $y(t)$ are displayed in solid lines, and the dashed lines indicate the weighted basis functions.

The choice of the basis function φ is crucial when determining the possible shapes that $\mathbf{r}(t)$ can reproduce[136] and the overall computational load of the segmentation algorithm[134]. For that reason, we use a function that has advantageous properties in terms of computational load and smoothness[130]. Moreover, this particular basis function has the property to allow the snake to perfectly describe ellipses. This property has been shown to be very important when delineating cross sections of cylindrical-like conduits and blob-like objects[130]. The explicit expression is given by

$$\varphi(t) = \frac{1}{1-\cos\frac{2\pi}{M}} \begin{cases} \cos\frac{2\pi|t|}{M}\cos\frac{\pi}{M} - \cos\frac{2\pi}{M} & 0 \le |t| < \frac{1}{2} \\ \frac{1}{2} - \frac{1}{2}\cos\frac{2\pi(3/2-|t|)}{M} & \frac{1}{2} \le |t| < \frac{3}{2} \\ 0 & \frac{3}{2} \le |t| \end{cases} \tag{4.10}$$

The energy function is defined such that the curve $\mathbf{r}(t)$ is attracted to image regions with high brightness gradient (here high level of Wg-Ptc expression). One issue is that a lack of fluorescent protein expression is typically observed along the outer boundary of the pouch in the region of the VP compartment, see **Figure 4.11A** for an example. To tackle this issue, convex configurations of $\mathbf{r}(t)$ are preferred during the optimization to prevent snakes to leak out from the pouch through low-intensity boundaries. Thus, a penalty term rewarding convex

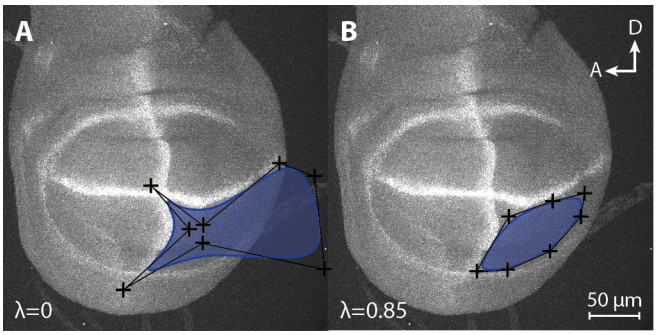


Figure 4.13: Illustration of the effect of λ on the detection of wing pouch compartments. **(A)** For $\lambda = 0$, the contribution of the shape energy is null and the snake tries to fit accurately the contour of the compartment. However, there is the risk that the snake would leak out from the compartment through low-intensity boundaries. **(B)** With $\lambda = 0.85$, convex solutions are preferred during the optimization, thus preventing the snake to leak out from the compartment. The effective contour of the snakes is displayed as a solid line enclosing a blue region. The '+' elements represent the control points of the model and their location define the effective contour of the snakes.

shapes is added to the snake energy now given by

$$E_{\text{snake}} = (1-\lambda)\, E_{\text{edge}} + \lambda\, E_{\text{convex}} \tag{4.11}$$

where $0 \leq \lambda < 1$ is a trade-off parameter that regulates the contribution of the convex penalty E_{convex} and the edge-based energy E_{edge}. **Figure 4.13** shows the effect of the parameter λ on the shape of the snake model after optimization. In addition, the importance of the convex penalty term E_{convex} is highlighted while identifying the shape of a wing pouch compartments. For $\lambda = 0$, E_{convex} is discarded and the snake focus on fitting accurately fluorescent boundaries (**Fig. 4.13A**). However in that configuration, there is the risk that the snake would leak outside of a compartment through low-intensity boundaries, that's why the convex penalty term has been introduced. For $\lambda = 0.85$, the priority of the snake becomes the conservation of a convex shape over fitting accurately fluorescent protein expression (**Fig. 4.13B**). It is important to note at that stage that the final model of the *Drosophila* wing pouch structure is reconstructed from the information provided by multiple detection modules (**Section 4.2.8**). Therefore it is definitively acceptable to obtain compartment models that do not match perfectly the contour of the pouch compartments. This actually provides a very good example of the overall robustness achieved inherent to the modular design of our unsupervised morphological structure detection method.

Traditionally, maps with the gradient information derived from the image are used to guide snakes to the actual contours of the object[128,129]. Here, we consider an energy that has the advantage of penalizing situations where the orientation of the curve (clockwise or counter-clockwise) is inconsistent with the boundary of the object to segment[138]. This property is very

important when the images contain more than one object to segment which is the case in our application.

As mentioned above, the energy term E_{snake} favors convex configurations of $\mathbf{r}(t)$. We integrate that information by minimizing the distance of $\mathbf{r}(t)$ to an elliptic fit to $\mathbf{r}(t)$. Any departures from a perfect ellipse would have a non-negative contribution to E_{snake}. Since all ellipses in the plane are affine transformations of the unit circle, we describe the ellipse $\mathbf{r}^{\text{p}}(t)$ as an affine transformation of a reference unit circle denoted by $\mathbf{r}^{\text{ref}}(t)$. Formally, we have

$$\mathbf{r}^{\text{p}}(t) = \mathbf{A}\,\mathbf{r}^{\text{ref}}(t) + \mathbf{b} \tag{4.12}$$

where the linear transformation $\mathbf{A}$ and the translation vector $\mathbf{b}$ of the affine transformation are determined by the minimization of

$$\int_0^1 \|\mathbf{r}(t) - \mathbf{A}\,\mathbf{r}^{\text{ref}}(t) - \mathbf{b}\|^2 \,\mathrm{d}t \tag{4.13}$$

Assuming that $\mathbf{r}(t)$ and $\mathbf{r}^{\text{ref}}(t)$ are described by the same basis function φ, we can equivalently restate the fitting problem using only the control points. This is possible thanks to the fact that our choice of φ guarantees that there exists a sequence of control points $\{\mathbf{c}[k]\}_{k\in\mathbb{Z}}$ so that circles and ellipses can be perfectly reproduced[130], and that φ generates a Riesz basis[136]. Thus, we define

$$\mathbf{c}^{\text{p}}[k] = \mathrm{A}\mathbf{c}^{\text{ref}}[k] + \mathbf{b} \tag{4.14}$$

and

$$\sum_{k=0}^{M-1} \|\mathbf{c}[k] - \mathrm{A}\mathbf{c}^{\text{ref}}[k] - \mathbf{b}\|^2 \tag{4.15}$$

where $\mathbf{c}^{\text{p}}$ and $\mathbf{c}^{\text{ref}}$ are the control points of $\mathbf{r}^{\text{p}}$ and $\mathbf{r}^{\text{ref}}$, respectively. Most of the computational efficiency of the method lies in this property since the possibility is offered to optimize a given continuous criterion using a small number of variables (i.e. the control points). More detailed information about this property is given in **Appendix D.2**.

4.2.6 Detecting the outer boundary of the wing pouch

This module takes as input the four compartment models inferred in **Section 4.2.5** to generate a parametric model of the outer boundary or contour of the *Drosophila* wing pouch delimited by the expression of Wingless.

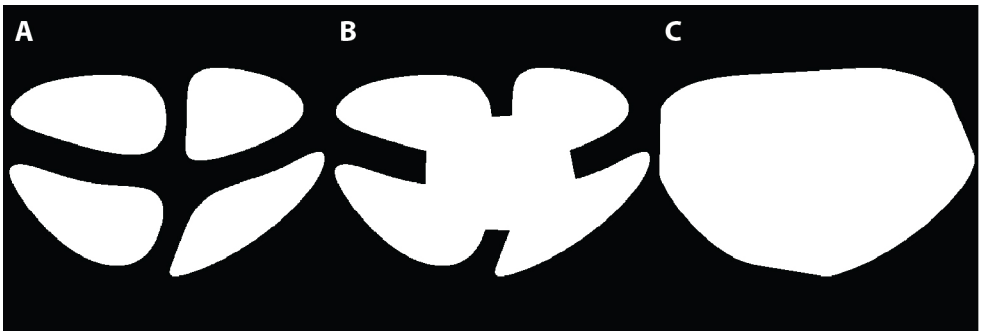


Figure 4.14: Detection and modeling of the wing pouch outer boundary. **(A)** Binary representation of the four compartments identified in **Section 4.2.5** using spline-based snakes. **(B)** The four compartments are aggregated to form a single connected component. **(C)** A standard boundary tracing algorithm[139] is applied to build a polygon enclosing entirely the structure, which is then convexified[140] before obtaining a *convex hull*. Finally, the contour of the convex hull is sampled and represented by a spline curve model.

First, we compute a binary image featuring the four spline-based snake models previously inferred **Section 4.2.5** (**Fig. 4.14A**) before aggregating them to obtain a single connected component by drawing a quadrilateral whose vertices are defined as the centers of mass of the snakes (**Fig. 4.14B**). A standard boundary tracing algorithm[139] is used to build a polygon enclosing entirely the structure, which is then convexified[140] to obtain a *convex hull* (**Fig. 4.14C**). Finally, the contour of the polygon is sampled before being described by the parametric spline curve model given by (4.9).

4.2.7 Detecting the A/P and D/V compartment boundaries (Part II)

The goal of this module is to infer an accurate model of the trajectory of the A/P and D/V boundaries. At that stage, useful information is already available from the detection modules applied previously including the location of the wing pouch center (**Section 4.2.3**), the direction of the A/P and D/V boundaries starting from the wing pouch center and given by the kite snake model (**Section 4.2.4**), and the outer boundary or contour of the wing pouch structure (**Section 4.2.6**).

One approach could have been to compute the equidistant lines between the four compartments previously identified and shown in **Fig. 4.14A**. However, this method has two main drawbacks. First, the convex penalty term in (4.11) prevents snakes leaking outside of their compartment while inhibiting precise fitting of the fluorescent boundaries, which is not an issue for our detection method as discussed in **Section 4.2.5**. Thus, computing equidistant lines from roughly detected compartments would not provide accurate models of the trajectory of the A/P and D/V boundaries. Moreover, this approach can simply not be applied in case one of the spline-based snakes leaks into another compartment.

Fluorescent boundary tracker

The method we propose is a mutli agent-based optimization technique that we developed and which simulates the behavior of line following robots (**Fig. 4.15**). The design of the agent model is similar to the one that we developed for the local optimization of the wing pouch center (**Fig. 4.5**). Each agent starts from the wing pouch center and then moves towards one of the four directions given by the segments of the kite snake model defined in **Section 4.2.4**. At each iteration, an agent moves using a predefined step-size and a combination of translation and rotation operators to align itself on the fluorescent boundary. The fluorescent boundaries of interest are defined by the expression of Wingless along the A/P and D/V boundaries inside the wing pouch. The iterative algorithm continues until the wing pouch outer boundary detected in **Section 4.2.6** is met by the agent. Finally, the A/P and D/V trajectories are parameterized using the samples generated by the agents using a spline curve model.

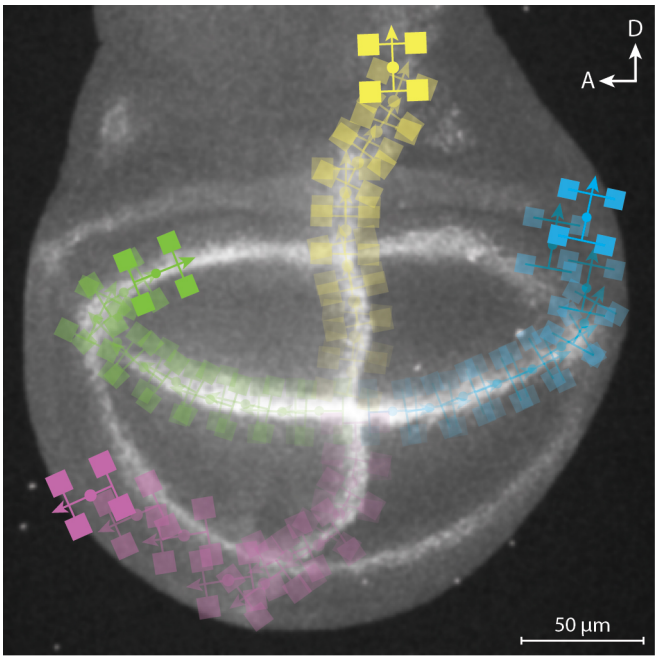


Figure 4.15: Parametrization of the A/P and D/V boundary trajectories using fluorescent boundary trackers. To accurately identify the A/P and D/V boundary trajectories, we developed a model of agents whose behavior is similar to the one of line following robots and is directly inspired from the local optimizer developed to detect the intersection of the A/P and D/V boundaries (**Fig. 4.5**). Here, we also show the trajectories that the agents would follow outside of the wing pouch to illustrate in more detailed the behavior of such agents. However, only the trajectory inside the wing pouch is obtained as the tracker automatically stops when it encounters the outer boundary of the pouch previously identified (**Section 4.2.6**).

4.2.8 Wing pouch structure construction

The information provided by the precedent detection modules is integrated into a single parametric model that describes the structure of the *Drosophila* wing pouch. The model inferred includes a parametric description of the A/P and D/V compartment boundaries, and the outer boundary of the pouch.

Figure 4.16A shows the typical output of the proposed structure detection method as we implemented it in the open-source image processing toolkit that we developed called WingJ for unsupervised and systematic quantification of biological systems. The '+' elements represent the control points of the inferred model which can still be fine-tuned manually to increase the accuracy of the parametric description of the morphological structure of the *Drosophila* wing pouch (**Fig. 4.16B**). The number of controls points can be modified before and after the inference process to achieve a more accurate description of the target structure, for instance. The independent '+' element visible in the top part of **Figure 4.16A** represents the center of mass of the *Drosophila* wing disc directly computed from the Wg-Ptc MIP image. This point is used in **Section 4.2.9** to detect the orientation of the wing, i.e. to infer the anterior, posterior, dorsal, and ventral sides of the model (APDV orientation).

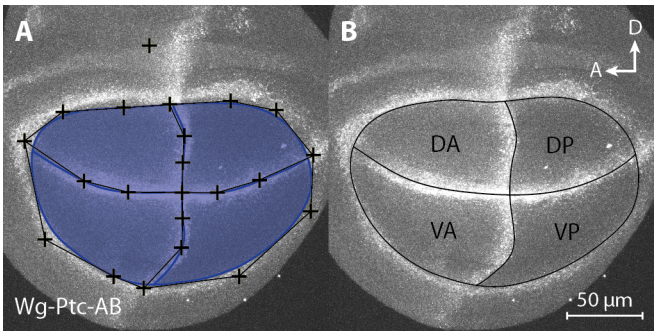


Figure 4.16: Construction of a parametric model describing the morphological structure of the *Drosophila* wing pouch. (A) Information from multiple detection modules are integrated to reconstruct a parametric model of the wing pouch structure. **(B)** The accuracy of the model can always be increased by manual fine-tuning. This can be achieved intuitively by moving around the control points to modify locally the shape of the wing pouch model.

Notes about exporting structure model files from WingJ

The inferred model of the *Drosophila* wing pouch structure can be exported in XML (Extensible markup language), for instance for further analysis. XML files are text documents that store information in a structured way[141]. Actually, two different XML documents are generated when exporting a structure dataset using WingJ.

- **Structure model.** This file includes the parameters (the control points) of the wing pouch model which are required to reconstruct a structure in WingJ. This allows to easily

edit a model and repeat or resume previous experiments.

- **Structure measurements.** This file includes measurements from the structure model, for example the length of the A/P and D/V boundaries, the length of each half-boundary C-A, C-P, C-D and C-V where C represents the pouch center and A, P, D, and V are points on the contour of the pouch, the perimeter and area of each of the four compartments, etc. All the measurements are given in µm and μm^2. This is possible because WingJ automatically extracts the information about the image scale from the input image files.

The structure measurement data can further be analysed using the WingJ Matlab Toolbox which is also part of our image processing toolkit. To be more precise, the Matlab toolbox provides an intuitively library of predefined functions that can be used to easily generate plots and statistical tests from datasets exported using WingJ (see **Section 4.8**).

4.2.9 Inferring the orientation of the wing pouch structure model

In addition to what is shown in **Figure 4.16B**, the orientation of the pouch model must still be determined. On other words, it is not known yet which part of the model corresponds to the anterior, posterior, dorsal, and ventral sides. Here, we propose an algorithm to reliably infer the orientation of the wing pouch based on morphological features and geometric constructions.

We define the set of points $\{C, P_1, P_2, P_3, P_4\}$ from the parametric wing pouch model (**Section 4.2.8**) where C is the intersection of the A/P and D/V boundaries (wing pouch center), and P_1, P_2, P_3 and P_4 are the intersections of the A/P and D/V boundaries with the outer boundary of the pouch. We also define M as the center of mass of the wing disc which is directly computed from the Wg-Ptc input image. We then apply **Algorithm 4.1** to infer the identity of each compartments (DA, DP, VA, and VP). **Figure 4.17** illustrates the geometric constructions made by the algorithm.

The closest point to M among P_1, P_2, P_3 and P_4 corresponds to the dorsal direction, thus $P_1 \rightarrow D$ and $P_3 \rightarrow V$. The keystone of the algorithm comes from the observation that the A/P boundary (D-V axis) is always curved (even if only slightly) to form an arc whose bottom part points towards the posterior side of the wing. Three line segments $\overline{DV}$ (here $\overline{P_1P_3}$), $\overline{CP_2}$ and $\overline{CP_4}$ are then built. If $\overline{DV}$ intersects $\overline{CP_4}$ then $P_4 \rightarrow A, P_3 \rightarrow P$, otherwise $P_3 \rightarrow A, P_4 \rightarrow P$. This method as two principal advantages. First, the result of the inference is independent of any rotation operations that may have been applied to the wing (e.g. during image acquisition) because the algorithm only relies on the relative information shared by six points in space. Note that the algorithm still works for very slight curvatures of the A/P and D/V boundaries.

Algorithm 4.1: Inference of the *Drosophila* wing pouch orientation

Data: Points P_1, P_2, P_3, P_4, C (wing pouch center), M (wing disc center of mass)
Result: Points D, V, A, P
begin `/* infer A/P boundary (D-V axis) */`
  D ← **Get closest point to M** (P_1, P_2, P_3, P_4);
  V ← **Get second boundary extremity point** (D);
end
begin `/* infer D/V boundary (A-P axis) */`
  $\{A, P\}$ ← **Get non-attributed points** (P_1, P_2, P_3, P_4);
  $\overline{CA}$ ← **Segment** (C, A);
  $\overline{CP}$ ← **Segment** (C, P);
  $\overline{DV}$ ← **Segment** (D, V);
  if *intersection*$(\overline{CP}, \overline{DV})$ **then**
    $\{A, P\}$ ← **Swap points** (A, P);
  end
end

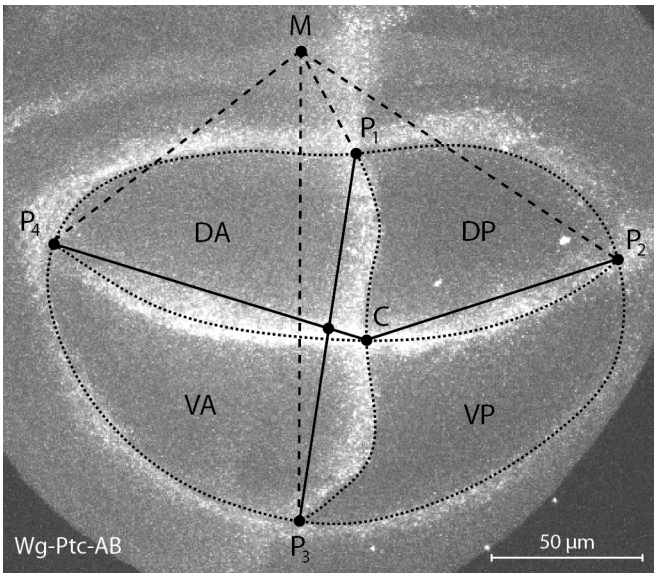


Figure 4.17: Inference of the orientation of the *Drosophila* wing pouch structure using *a priori* knowledge. From the model of the wing pouch structure obtained in **Section 4.2.8**, we define the set of points $\{C, P_1, P_2, P_3, P_4\}$ where C is the intersection of the A/P and D/V boundaries and P_1, P_2, P_3 and P_4 are the intersections of the A/P and D/V boundaries with the outer boundary of the pouch. We also define M as the center of mass of the wing imaginal disc. **Algorithm 4.1** is applied to infer the orientation of the model by matching the points $\{P_1, P_2, P_3, P_4\}$ to the anterior, posterior, dorsal, and ventral side of the wing. The principal benefit of the proposed orientation inference method is that it is independent of any rotation operations that may have been applied to the wing (e.g. during image acquisition) and that the method works fine even for slight curvatures of the A/P and D/V boundaries.

4.3 Quantification of expression in the *Drosophila* wing pouch

4.3.1 Background subtraction

In fluorescence microscopy, the level of background illumination is often nonuniform in space because scattered light from the medium or the specimen is typically inhomogeneous[142,143]. An *a priori* background correction can be performed by applying a local background subtraction technique, which consists in capturing the background image in the absence of the specimen before subtracting it from the specimen image[142,144]. Additional information are given in the image acquisition protocol described in **Appendix D.1.2**.

4.3.2 Mean intensity projection

Following **Appendix D.1.4**, we have at our disposal a collection of stacks of confocal fluorescence images, each of them containing information about the expression of a particular gene. We denote a stack of confocal images f. The standard coordinate system of an image stack f is defined by the triplet (x, y, z), where x and y are the coordinates of a single image (the origin of an image is traditionally located at the upper-left corner) and z describes the depth of the image stack. $f(x, y, z)$ is the value or brightness of the pixel located at (x, y, z) within the image stack. Here, we consider pixels taking grayscale values in $[0, 255]$ (8-bit encoding) as they were directly provided by the confocal microscope (**Appendix D.1.2**).

The average intensity projection (AIP) or mean projection is the 2D image computed by averaging the pixel values along the z-axis in between two slices located at $z = z_{\min}$ and $z = z_{\max}$. In opposition to computing the maximum intensity projection of the entire image stack to get a clear picture of the wing pouch structure before segmentation, we are selecting a reduced number of slices and performing the mean projection allowed us to reduce the noise as well as avoid the signal from the peripodial membrane. The selection of image slices to consider defined by $z_{\min}$ and $z_{\max}$, and the choice of the projection method are done directly in WingJ. Formally, the mean projection is defined as

$$\mathrm{AIP}\left(x, y\right) = \frac{1}{z_{\max} - z_{\min} + 1} \sum_{z=z_{\min}}^{z_{\max}} f\left(x, y, z\right) \tag{4.16}$$

Note that the expression measurements made below are taken over the continuously-defined image using a bilinear interpolation model and floating-point resolution for the intensity values. Namely, expression levels are measured directly from the interpolation model and quantified using floating-point numbers and not 8-bit integers. Therefore, the benefit of using an interpolation model is that the resolution of the measurement points is no longer restricted by the pixel discretization of the images.

4.3.3 Definition of the structure coordinate system

We described in **Section 4.2** a method to automatically infer a model of the morphological structure of the wing pouch. In addition to providing a wealth of morphological information, this model provides us with a valuable non-orthogonal coordinate system to measure gene and protein expression. The full potential of the selection of this coordinate system is described in **Section 4.4** where 2D and possibly 3D expression measurements taken from multiple wings are aggregated to generate a single and reliable quantitative description of the pouch. Here, we set the center of the coordinate system to the pouch center C previously identified as the intersection of the A/P and D/V boundaries. In this system, the x-axis follows the D/V boundary (positive values extending to the posterior side) and the y-axis follows the A/P boundary (positive values extending to the dorsal side).

4.3.4 Generating expression profiles

Expression profiles are widely used in the literature to report gene or protein expression levels along a 1D trajectory which actually corresponds to a 2D trajectory in the space of the image[50,145,146]. However, the accuracy of the measurements is often impeded by unnecessary approximations made in the method. For instance, measurement points are often restrained by the pixel discretization and encoding of the input image. Moreover, filters introduced to smooth expression profiles easily ignore the shape of the 1D trajectory. The method we describe here addresses all these issues and allows to generate accurate 1D expression profiles using our open-source image processing software.

We first compute a bilinear interpolation model from the mean intensity projection as previously mentioned. The model provides us with a continuous description of the projection on which expression measurements are made to overcome the limitation relative to the 8-bit encoding of the image. To give an example, **Figure 4.18A** shows the structure model identified for a 100-hour-old wing displayed over the mean intensity projection of Pmad-AB. Information about Brk-AB is also available for this wing (**Fig. 4.18B**). We decide to quantify the concentration level of these two proteins along the D/V boundary. The number of measurement points along the trajectory can be freely chosen as it is not restricted by the pixel discretization of the original image thanks to the interpolation model. Moreover, each point of the expression profile is obtained by averaging several measurements taken along a segment perpendicular to the trajectory at that point. The measurements are then weighted along this segment using a Gaussian distribution with standard deviation σ. **Figure 4.18C** reports the expression profiles obtained using our method for Pmad-AB and Brk-AB.

Furthermore, trajectories can be generated along translated versions of the A/P and D/V boundaries. The first image in **Figure 4.19A** shows the trajectory obtained by translating the D/V boundary along 90% of the length of the C-D axis (trajectory following the A/P boundary that starts from the wing pouch center C and ends at the intersection of the A/P boundary

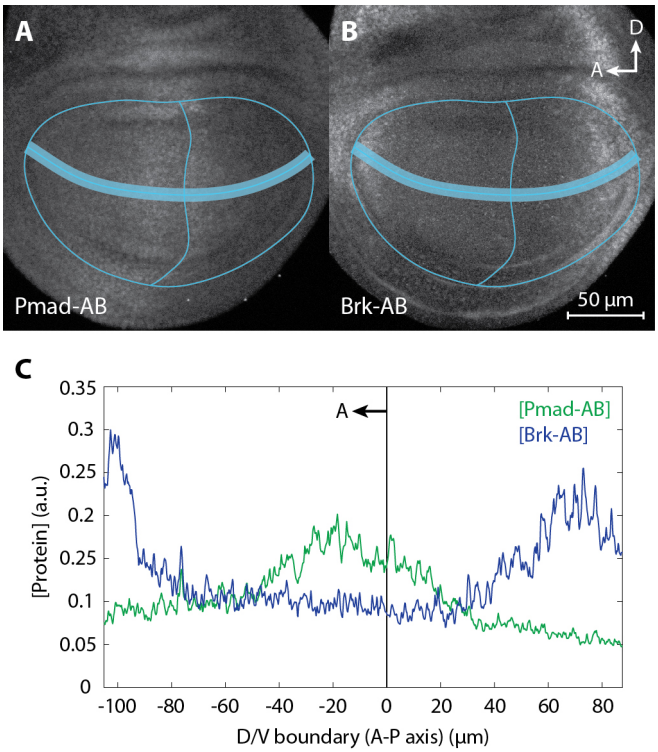


Figure 4.18: Quantification of Pmad and Brk expression profiles along the D/V compartment boundary. (A-B) The model of the wing pouch structure identified in **Section 4.2** (in blue) provides a valuable non-orthogonal coordinate system for measuring gene expression from fluorescence intensity. The blue region along the D/V boundary shows where expression is measured. **(C)** Expression level of Pmad (in green) and Brk (in blue). The x-axis of the coordinate system is defined by the inferred trajectory of the D/V boundary with $x = 0$ corresponding to the point where the A/P and D/V boundaries intersect (negative values extending to the anterior side).

and the dorsal part of the wing pouch outer boundary). In a similar way, the first image in **Figure 4.19B** shows the trajectory obtained by translating the A/P boundary along 90% of the length of the C-A axis (trajectory following the D/V boundary that starts from the wing pouch center C and ends at the intersection of the D/V boundary and the anterior part of the wing pouch outer boundary). The parts of the trajectories falling outside the wing pouch model are trimmed.

4.3.5 Generating circular expression maps

We have introduced above a method to accurately quantify expression profiles along 1D spatial trajectories, which are generated using the coordinate system derived from the parametric model of the wing pouch structure identified in **Section 4.2**. The advantage of using a coordinate system that is shared by multiple wings, despite obvious variations in their morphology, is that it enables the direct comparison of expression measurements taken at the same relative

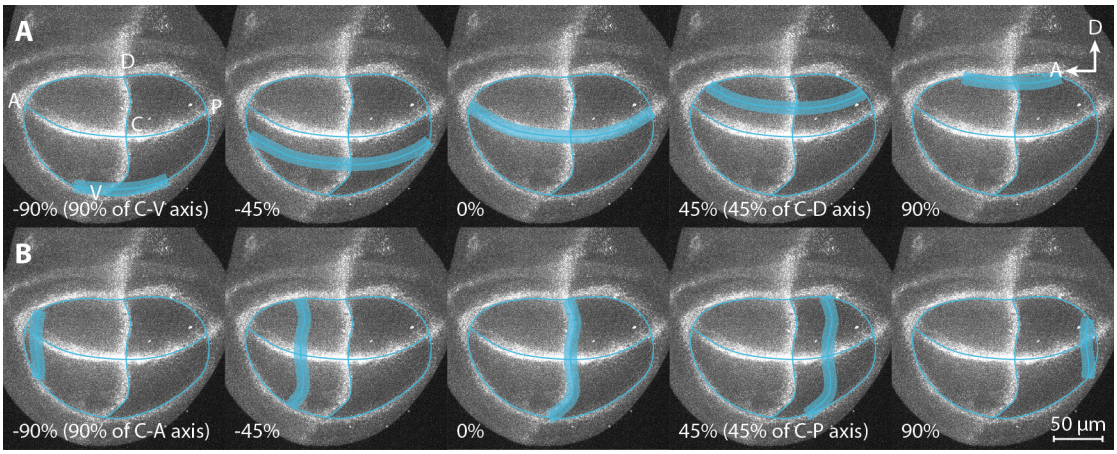


Figure 4.19: Expression quantification along translated versions of the inferred A/P and D/V boundary trajectories. The wing pouch structure model identified in **Section 4.2** (in blue) provides a valuable non-othogonal coordinate system for measuring expression levels at a given relative location in multiple wings. Expression profiles can be measured along 1D trajectories that are translated instances of the reference boundaries **(A)** A/P and **(B)** D/V. For instance, the top-left image shows the trajectory obtained by translating the detected D/V boundary along 90% of the length of the half-boundary C-D.

locations in different wings.

While expression profiles already provides valuable information for modeling gene and protein expression[50,145–147], we decided to extend the approach to the generation of 2D expression maps. We give a formal description of a method we developed for generating expression maps and thus enable the modeling and study of gene expression in 2D representations of biological systems (organ system or body system). Note that our approach can be fairly easily extended to generate 3D expression maps.

Because the morphology of two biological systems will always differ, we sample gene expression using the wing pouch coordinate systems (**Section 4.3.3**) before generating a circular expression map. This representation is *scaleless* and provide a convenient description enabling the integration of data from several organisms.

First, we define a coordinate grid based on the wing pouch coordinate systems. As a reminder, the inferred model of the wing pouch structure provides a parametric description of the A/P and D/V boundary trajectories, and the trajectory of the pouch outer boundary. The origin of the coordinate system formed by the A/P and D/V boundaries is defined by their intersection, which we named the wing pouch center. The grid illustrated **Figure 4.20** takes inspiration from the parameterization of the surface of Earth using *parallels* and *meridians* (latitude and longitude). We define the four cardinal point of the grid as $\mathrm{N} = (n_x, n_y)^\mathrm{T}$, $\mathrm{S} = (s_x, s_y)^\mathrm{T}$, $\mathrm{E} = (e_x, e_y)^\mathrm{T}$, and $\mathrm{W} = (w_x, w_y)^\mathrm{T}$. The center of the grid is aligned on the wing pouch center and is labelled $\mathrm{C} = (c_x, c_y)^\mathrm{T}$.

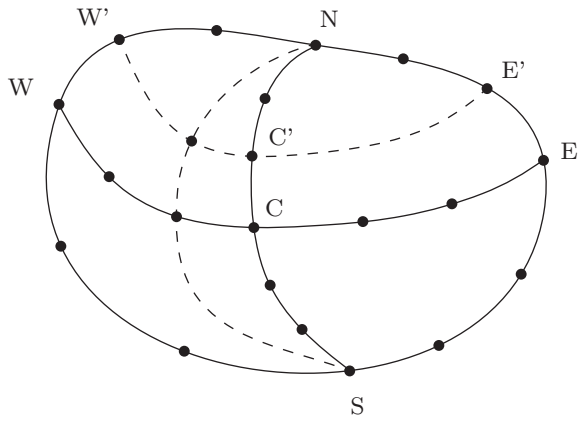


Figure 4.20: Schematic representation of the coordinate grid defined for producing 2D expression maps. This grid is inspired from Earth coordinate system grid and is based on the parametric model of the wing pouch structure identified in **Section 4.2**. We denote N, S, E, and W as the four cardinal points. The origin of the grid is aligned on the wing pouch center C defined as the intersection of the A/P and D/V boundaries. The orientation of the grid depends on the choice of the A/P or D/V boundary for the equator (here the equator is set along the D/V compartment boundary). The grid is defined continuously, thus expression maps can be generated for any arbitrary resolution.

Affine transformations $\mathcal{T}$ are applied to the equator (the curve that joins the points W, C, and E) to construct the *parallels* of the grid. $\mathbf{WC}(t)$ is defined as the parametric curve starting at W and ending at C, and $\mathbf{CE}(t)$ is the curve joining C to E. The new instance of the equator after transformation is given by

$$\mathbf{WC}'(t) = \mathcal{T}\{\mathbf{WC}(t)\} \tag{4.17}$$
$$= \mathbf{A}\,\mathbf{WC}(t) + \mathbf{b} \tag{4.18}$$
$$\mathbf{CE}'(t) = \mathcal{T}\{\mathbf{CE}(t)\} \tag{4.19}$$
$$= \mathbf{A}\,\mathbf{CE}(t) + \mathbf{b} \tag{4.20}$$

where

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \tag{4.21}$$

is a 2-by-2 matrix and $\mathbf{b}$ is a 2-by-1 vector of the affine transformation $\mathcal{T}$. Because we constrain the new instance of the equator to interpolate the three points W′, C′, and E′, the affine transformation $\mathcal{T}$ is uniquely determined as long as W, C, and E are not collinear. Formally, we impose $\mathrm{W}' = \mathcal{T}\{\mathrm{W}\}$, $\mathrm{C}' = \mathcal{T}\{\mathrm{C}\}$, and $\mathrm{E}' = \mathcal{T}\{\mathrm{E}\}$. Thus, the coefficients of the matrix $\mathbf{A}$ are given by

$$
\begin{aligned}
a_{11} &= \frac{(w_y - e_y)(w'_x - c'_x) - (w_y - c_y)(w'_x - e'_x)}{(w_y - e_y)(w_x - c_x) - (w_x - e_x)(w_y - c_y)} \\
a_{12} &= \frac{(w_x - c_x)(w'_x - e'_x) - (w_x - e_x)(w'_x - c'_x)}{(w_y - e_y)(w_x - c_x) - (w_x - e_x)(w_y - c_y)} \\
a_{21} &= \frac{(w_y - e_y)(w'_y - c'_y) - (w_y - c_y)(w'_y - e'_y)}{(w_y - e_y)(w_x - c_x) - (w_x - e_x)(w_y - c_y)} \\
a_{22} &= \frac{(w_x - c_x)(w'_y - e'_y) - (w_x - e_x)(w'_y - c'_y)}{(w_y - e_y)(w_x - c_x) - (w_x - e_x)(w_y - c_y)}
\end{aligned}
$$

and the vector **b** is computed as follows:

$$\mathbf{b} = \frac{1}{2}\left(\mathrm{W}' + \mathrm{C}' - \mathbf{A}(\mathrm{W} + \mathrm{C})\right) \tag{4.22}$$

The points W′, C′ and E′ are selected in such a way that they equally divide the curves running form the cardinal point N to W, C and E, respectively:

$$\frac{\text{length}(\mathrm{W}', \mathrm{N})}{\text{length}(\mathrm{W}, \mathrm{N})} = \frac{\text{length}(\mathrm{C}', \mathrm{N})}{\text{length}(\mathrm{C}, \mathrm{N})} = \frac{\text{length}(\mathrm{E}', \mathrm{N})}{\text{length}(\mathrm{E}, \mathrm{N})} \tag{4.23}$$

In the same way, the affine transformation $\mathscr{T}$ defined by (4.21) is applied to compute the *meridians* (vertical lines in **Fig. 4.20**). A parametric grid is then obtained by combining parallels and meridians together. The nodes of the grid, defined as the intersections of the parallels and meridians, indicate where expression is measured. It is important to note that since the grid is parametric, its resolution can be freely chosen.

Two different expression maps are generated when placing the equator along the A/P or D/V boundary, respectively. **Figure 4.21** shows the two different grids obtained as well as the resulting Wg-Ptc circular expression maps. For example, the grid displayed in **Figure 4.21A** is obtained by placing the equator along the A/P boundary. Wg-Ptc expression is then sampled at each grid node before generating the expression map shown in **Figure 4.21C**. We observe in **Figure 4.21C** that the Wg-Ptc expression along the A/P axis is well conserved, however remote regions are affected by artifacts. The same phenomenon also affects the maps of Earth. Namely, the issue comes from the fact that the grid warping produces artifacts on the regions close to the *poles* due to oversampling in the grid. Nevertheless, the circular expression map is only a temporary representation used later to aggregate expression data from multiple wings, before being wrapped back on a pouch structure model.

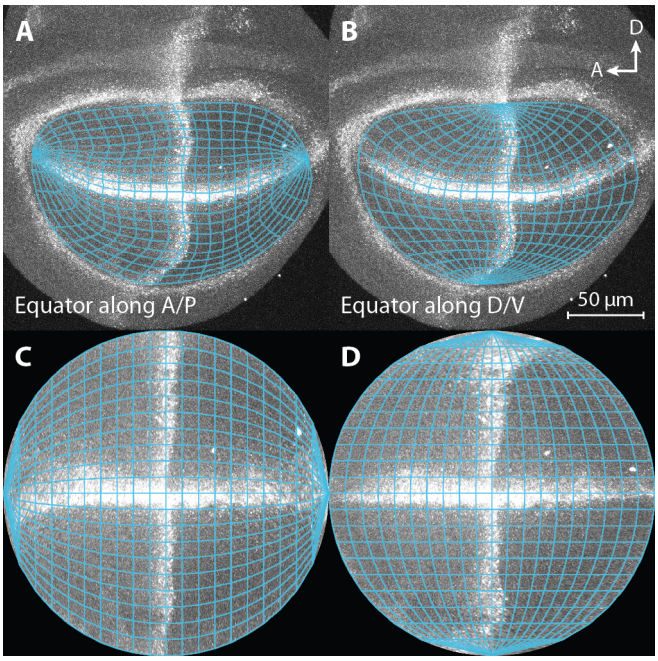


Figure 4.21: Illustration of the choice of the grid equator for generating circular expression maps. The wing pouch structure model identified in **Section 4.2** and shown here in blue is derived to define a grid inspired from the parameterization of the surface of Earth. The nodes of the grid determine where expression measurements will be sampled. Two different grids are produced depending on the choice of the main boundary (A/P or D/V) along which expression will be measured and analyzed. Here, the equator of the grid is set along **(A)** the A/P and **(B)** D/V boundary. **(C-D)** By looking at the Wg-Ptc circular expression maps generated, one can observe that the expression is mainly conserved along the A/P and D/V boundary.

4.4 Integration of structure and expression models

In **Section 4.2**, we have formally describe a method for unsupervised detection and modeling of the *Drosophila* wing pouch structure from Wg-Ptc confocal fluorescence images. The principal achievement is the generation of a parametric model that quantitatively describes the morphological structure of the pouch including the A/P and D/V boundary trajectories, and the outer boundary of the pouch. We then defined a coordinate system based on the inferred structure model to quantify gene and protein expression in confocal fluorescence images (**Section 4.3.4**). More specifically, we proposed a method for accurate generation of gene expression profiles along 1D trajectories. Because the information in 1D expression profiles is rather limited despite being largely used in the literature[50,145,146], we developed a method based on the previously defined coordinate system to measure expression in the entire space of the inferred structure model. Using this approach, we generated circular, scaleless 2D expression maps whose strength lies in enabling the relative and comparative analysis of gene expression in several wings (**Section 4.3.5**).

In the subsequent sections, we introduce a method to combine several independent models

previously identified for many *Drosophila* wings. The result is the generation of an aggregated model that provides a more reliable quantitative description of the morphology of the wing (**Section 4.4.1**). We then describe a procedure to aggregate 1D expression profiles and 2D expression maps (**Sections 4.4.2** and **4.4.3**). Finally, we integrate both structure and expression data to generate a single and robust quantitative description of the wing pouch.

4.4.1 Integrating structure models

In the first place, we start by observing that models of the *Drosophila* wing pouch structure inferred in **Section 4.2** are likely to all have a different orientation in addition to be located to different areas in each input image. Before being able to integrate them in order to generate a single, representative model of the wing pouch structure, we need to quantitatively describe their orientation before applying the operations required to have all of them in the same orientation.

In **Section 4.3.3**, we defined a non-orthogonal coordinate system based on the inference of the A/P and D/V boundary trajectories. We then define the *canonical orientation* of a structure model where dorsal and anterior sides are directed towards the top and left sides of the image. More precisely, the canonical orientation corresponds to a *clockwise* structure whose segment $\overline{\text{CD}}$ points vertically upwards and whose points D, P, V, and A appear in clockwise order in the space of the image. **Figure 4.22** illustrates the different orientation configurations that a structure model can take. The canonical orientation is shown in **Figure 4.22A** and examples of clockwise models are given in **Figures 4.22A** and **4.22D**. At first, we require all the models to be in clockwise orientation but not necessarily in canonical orientation. To achieve this, a horizontal reflection is applied to **Figure 4.22B**-like structure models and a vertical reflection is applied to **Figure 4.22C**-like models.

Once all the structure models are in clockwise orientation, we pick arbitrarily one of them to become the reference structure and place it in the *canonical* orientation. We achieve this by rotating the selected structure so that its segment $\overline{\text{CD}}$ points vertically upwards. As a remainder, C is the wing pouch center, which we defined as the intersection of the A/P and D/V boundaries, and D is the intersection of the A/P boundary with the dorsal part of the structure outer boundary. The remaining models have then their center of gravity G aligned on the center of gravity of the reference structure. It is important to note that the center of gravity does not correspond to the wing pouch center. Before proceeding effectively to the integration of the structure models, we still need to align the orientation of the models on the orientation of the reference structure, which translates into the minimization of a function that takes as input the angles shown in **Figure 4.23**. Formally, we define

$$f(\theta) = (\theta_{GD} - \theta)^2 + (\theta_{GP} - \theta)^2 + (\theta_{GV} - \theta)^2 + (\theta_{GA} - \theta)^2 \tag{4.24}$$

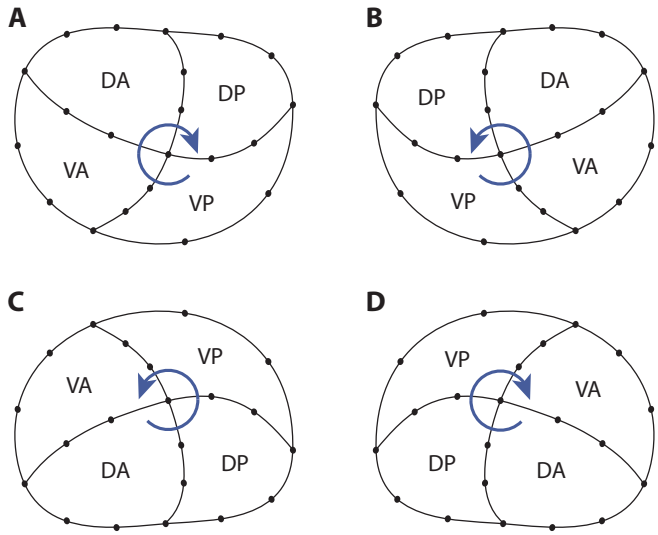


Figure 4.22: Orientation configurations of the wing pouch structure model. (A) The canonical orientation is defined as the orientation of the structure coordinate system described in **Section 4.3.3**. Dorsal and anterior sides of the inferred structure model must be directed towards the top and left sides of the image, respectively. The canonical orientation requires the points D, P, V, and A to appear in clockwise order in the space of the image. The remaining possible orientations of inferred structure models are given in **(B)** counterclockwise (horizontal reflection required to be clockwise), **(C)** counterclockwise (vertical reflection required to be clockwise), and **(D)** clockwise.

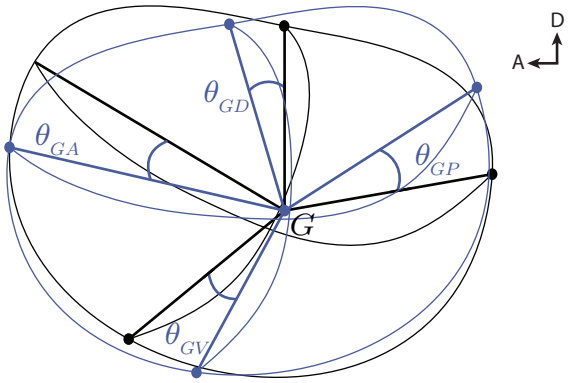


Figure 4.23: Optimization of the alignments of two structure models. Averaging structure models requires first to optimize the respective alignements of the individual structures. One structure is arbitrarily selected and placed in the canonical orientation (here in black). A second structure is selected and its center of gravity is aligned on the one of the first structure. An optimization process is then applied to minimize (4.25) which involves the angles θ_{GD}, θ_{GP}, θ_{GV} and θ_{GA}, and thus optimize the respective alignment of the two structures. The same procedure is then applied to the remaining structure models.

before minimizing

$$\frac{\mathrm{d}f}{\mathrm{d}\theta}(\theta) = 0 \tag{4.25}$$

where the angles θ_{GD}, θ_{GP}, θ_{GV}, and θ_{GA} result from the misalignment of the segments $\overline{\text{GD}}$, $\overline{\text{GP}}$, $\overline{\text{GV}}$, and $\overline{\text{GA}}$ of two different structure (**Fig. 4.23**). (4.25) is quadratic, and a closed solution can be obtained by computing its derivatives and equating it to zero.

In practice, the structure models may have been generated using different numbers of control points. This depends on the choice of the experimenters as higher numbers of control points usually allow to describe more accurately the structure of a body or organ system while increasing its complexity (**Section 4.2.8**). Here, we require all the structure models to be described using the same number of control points. Because removing control points often leads to the deformation of the structure and thus to a loss of information, we increase the number of control points of each model so that they match the highest number featured by a model in the set.

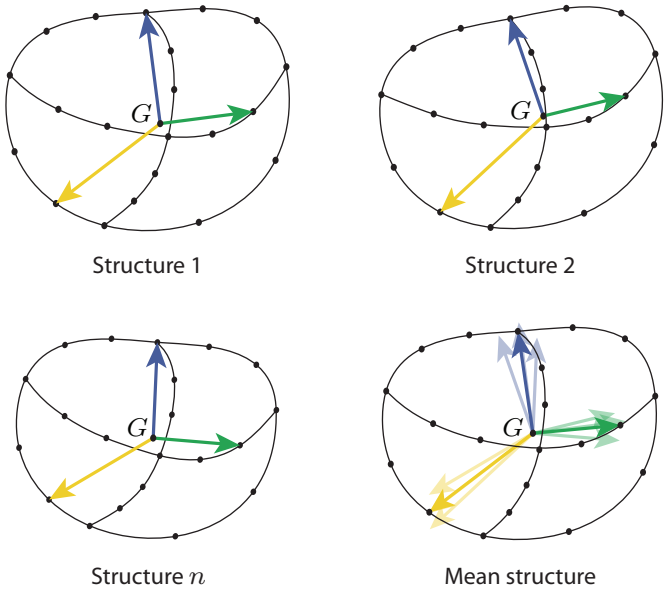


Figure 4.24: Integration of inferred structure models. The aggregation of structure models requires 1) to place each model in clockwise orientation, 2) to align the center of gravity G of the models, 3) to optimize their relative alignment by minimizing (4.25), and 4) to set the number of control points of each structure to the highest number featured by a model of the set. The aggregated structure is built from the center of gravity G. To compute the aggregated version of a given control point (black dots), we produce a set of vectors starting each from the point G of their model and ending at the location of the control point. The corresponding control point in the aggregated model is then obtained by computing the *resultant* of the vector set. We repeat this procedure to every control points before using the new points to parameterize the aggregated structure model (**Sections 4.2.5** and **4.2.8**). A concrete example generated using WingJ is given in **Figure 4.25**.

Figure 4.24 illustrates the averaging of several structure models parameterized by their control points (black dots). Note that now the center of gravity G of each model are aligned on the same point (it does not matter where this point is in the image space). The generation of

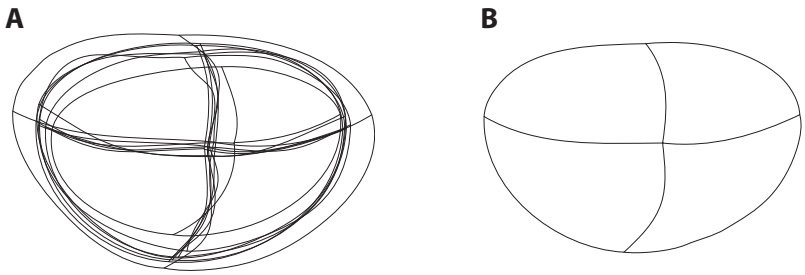


Figure 4.25: Integration of *Drosophila* wing pouch structure models in WingJ. (A) Individual model of seven *Drosophila* wing pouches inferred by WingJ (**Section 4.2**). **(B)** The individual models previously inferred and exported to files are provided to WingJ for generating a single and reliable quantitative description of the *Drosophila* wing pouch structure. Note that the aggregated models are always produced in the canonical orientation.

aggregated structures is based on the aggregation of the control points of individual models, which entirely define the shape of the structure model (**Sections 4.2.5** and **4.2.8**). For each control point, we produce a set of vectors starting from the point G and ending at the locations of the given control point in the different individual models. The resultant of this set of vectors then defines the position of the new control point in the aggregated model. This operation is repeated for every control points (including the control point defining the wing pouch center C) in order to obtain a complete parameterization of the aggregated model. **Figure 4.25** shows a concrete example of the aggregation of several *Drosophila* wing pouch structure models using WingJ. Note that the aggregated models are always produced in the canonical orientation.

We have described a method to generate a single quantitative description representative of several individual structure models previously inferred. We have shown in **Figure 4.25B** the output of this method when applied to the individual models of the structure of seven *Drosophila* wings. If the averaged model already provides a more accurate and reliable description of the morphology of a body or organ system than any individual model taken separately, there is still an important information that is not included in **Figure 4.25B**. Namely, this concerns how much variation or dispersion exists from the mean structure model. When computing the resultants of the control points defining the mean structure, we measure the length of the individual vectors associated to a given control points and evaluate their standard deviation σ. By adding the different σ to the length of their respective resultant, we create a new set of control points used to generate a closed B-spline curve that follows the outer boundary of the mean structure model extended by the computed standard deviation. **Figure 4.28** shows an example where this curve provides direct visual feedback on the variability of the individual structure models used to build the mean model.

4.4.2 Integrating expression profiles

We described in **Section 4.3.4** an approach to accurately quantify gene and protein expression levels along a trajectory. Here, we explain how the expression profiles from many individual experiments can be integrated to generate more representative and meaningful expression profiles. Similar profiles have been used to reverse engineering the gap gene network responsible of the *Drosophila* embryo patterning[39,148] or to study scaling in gene expression domains[50].

Individual expression profiles from different experiments are expected to not have the same length along the spatial axis (x-axis) and their respective number of samples or measurements points may not match (**Fig. 4.26**A). Using WingJ, users are free to choose between using a fixed number of measurement points or defining the distance (e.g. in µm) between two samples. For each profile, we evaluate the length of the negative and positive parts on the x-axis, respectively $x_{i,min}$ and $x_{i,max}$ where i is the index of the profile. From the definition of the structure coordinate system (**Section 4.3.3**), the negative part on the x-axis corresponds to the anterior or ventral side of the structure depending on the reference boundary selected (A/P or D/V) to generate the trajectory along which expression levels have been measured (**Section 4.3.4**). If the trajectory has been chosen parallel to the D/V boundary, $x = 0$ is where the A/P boundary intersects the trajectory. The length of the negative and positive parts on the x-axis of the mean expression profile are given by

$$\bar{x}_{min} = \frac{1}{n}\sum_{i=1}^{n} \bar{x}_{i,min} \tag{4.26}$$

$$\bar{x}_{max} = \frac{1}{n}\sum_{i=1}^{n} \bar{x}_{i,max} \tag{4.27}$$

where n is the number of individual profiles to integrate. Then, the space defined by $[\bar{x}_{min},\bar{x}_{max}]$ is evenly divided before interpolating each profile along these points. Then, sample points are selected along the x-axis where the expression profiles are averaged. Note that the number of sample points should usually be smaller than the number of interpolation points. **Figure 4.26B** reports the mean expression profiles computed from the individual profiles shown in **Figure 4.26A**. The error bars correspond to the standard error, the number of interpolation points is set to 200, and the number of sample points to 50.

4.4.3 Integrating circular expression maps

We have described in **Section 4.3.5** how we quantify gene and protein concentration levels in the space defined by the parametric description of the morphology of the wing pouch or any other body or organ system considered. To achieve this, we derived the structure model to formally define a non-orthogonal coordinate system for sampling expression levels in a meaningful way before combining the measurements to build a circular expression map. We

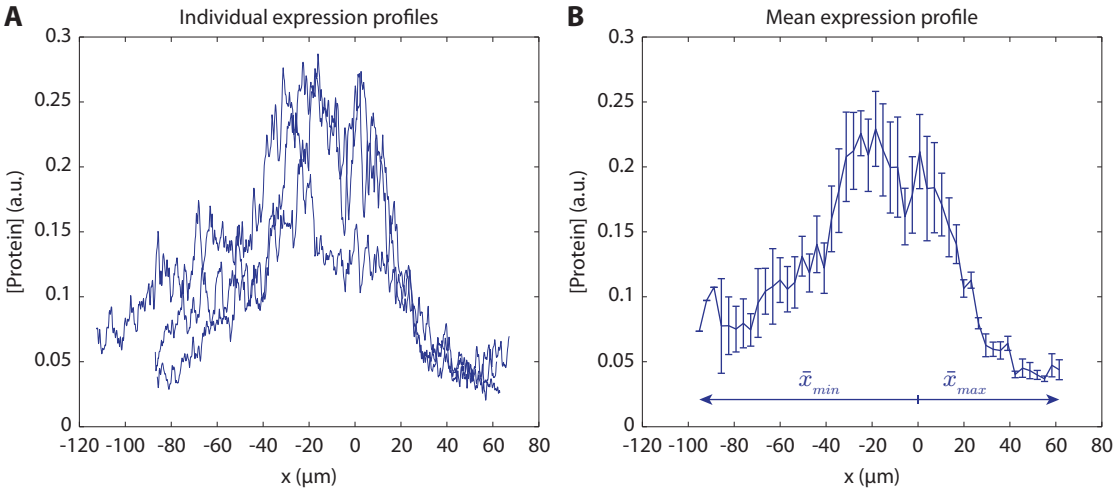


Figure 4.26: Integration of individual expression profiles to obtain meaningful and usable profiles. (A) Individual expression profiles exported from WingJ. **(B)** The length of the mean negative and positive distances $\bar{x}_{min}$ and $\bar{x}_{max}$ are computed from the x extremities of the individual profiles. Because each profile may not have been sampled at the same x locations, the space defined by $[\bar{x}_{min},\bar{x}_{max}]$ is evenly divided before being used to resample each individual profile. Then, the individual profiles have been interpolated along $[\bar{x}_{min},\bar{x}_{max}]$ using 200 points.

apply this method to quantify the concentration levels of several genes and proteins in many wings imaged under similar conditions. Here, we describe an approach to integrate multiple circular expression maps (**Fig. 4.27A**) into a single and robust mean circular expression map (**Fig. 4.27B**), which are later wrapped on mean structure models.

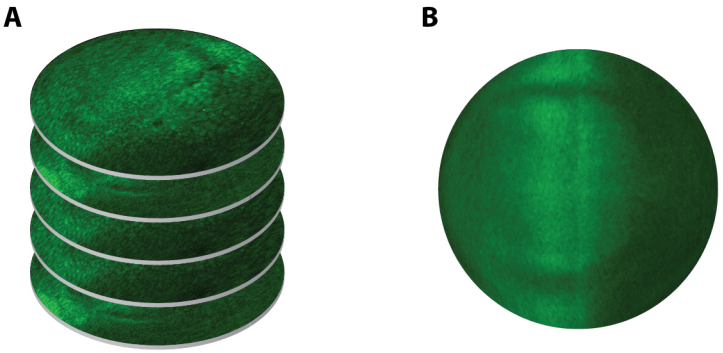


Figure 4.27: Generation of mean expression maps from individual disc-shaped maps. (A) The quantification of gene expression requires confocal fluorescence images (shown here for Pmad-AB measured in *Drosophila* wing) and a parametric model of the structure of the system considered. A coordinate system is defined based of this model, which is then used to sample expression and build a circular expression map. **(B)** The mean intensity projection is computed from individual maps to generate a reliable mean circular expression map. All the expression maps generated using WingJ are given in the canonical orientation.

Because the morphology of each wing is different, it would not have been possible initially to generate a representative and quantitative description of gene expression from many wings. However, circular expression maps provide a convenient way to process expression data. For example, we can easily compute an average intensity projection of the individual expression maps to generate a mean expression map much more reliable and accurate than any single individual map (**Fig. 4.27B**). Formally, a point in the mean circular expression map $\bar{e}(x,y)$ is given by

$$\bar{e}(x,y) = \frac{1}{n}\sum_{i=1}^{n} e_i(x,y) \tag{4.28}$$

where $e_i(x,y)$ is the i$^{\text{th}}$ individual expression map, and x and y the coordinates of a measurement point.

4.4.4 Generating structure and expression aggregated models

A single image of a biological organism or organ is often selected and included in publications for illustration purposes. However, important information is missing such as the variability in shape of the system or variability in gene and protein expression level. Therefore, we propose a method to integrate the morphological and expression data collected from multiple wings into a single, reliable and robust quantitative description.

Here, the approach consists first in quantifying the structure and expression level in individual wings. These data are then integrate to obtain a mean structure model and a mean circular expression map for each gene or gene product available. Information about the variability of the morphology or gene expression are also computed at the same time as computing the mean models. The last step consists in wrapping the circular expression map on the mean structure model. The wrapping is performed in two steps. First, we create a sampling grid adapted to the mean structure model following the algorithm described in **Section 4.3.5**. This grid can be interpreted as a continuous mapping $\mathscr{M} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that transforms any point in the structure $\mathbf{p}$ to a point $\mathbf{p}' = \mathscr{M}(\mathbf{p})$ on the circle grid associated to the circular expression map (**Fig. 4.21**). Then for any grid-point $\mathbf{p}$ on the sampling grid, we measure the value of the expression on $\mathscr{M}(\mathbf{p})$ over the mean expression map. This value is put back on the closest pixel location within the space defined by the mean structure model.

Figure 4.28 illustrates the final representation of the wing pouch obtained. This quantitative description of the wing has been reconstructed from the morphological and expression information collected in eight *Drosophila* wings imaged 90 hours AEL. The three channels of the image represent the concentration levels of Brk-AB (red), Dad-GFP (green), and Omb-AB (blue). The dashed line provide information about the standard deviation of the size of the wing pouch. Finally, we suggest that the use of such quantitative descriptions would be suitable in research and for illustration purpose in publications. In addition to provide an

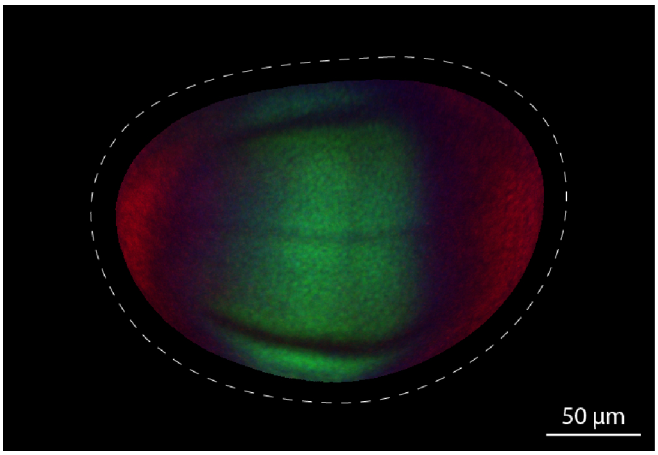


Figure 4.28: Quantitative description of the *Drosophila* wing pouch generated from eight wings using our method. First, structure and expression information is quantified in each wing available. The collected data are then integrated in order to obtained a mean structure model of the pouch and mean circular expression maps for each gene and protein expression available in the stacks of confocal images. Expression maps are then wrapped on the mean structure model to obtain a single, reliable and robust quantitative description of the wing pouch. Here, Brk-AB is in red, Dad-GFP in green, and Omb-AB in blue. The dashed line represents the standard deviation of the size of the pouch. Finally, this result is achieved through a fully automated process that only requires stacks of confocal fluorescence images.

appealing visual representation, quantitative descriptions generated using our method can be used directly as input for the reverse engineering of gene network models, for instance.

4.5 Unsupervised cell nuclei detection and segmentation

A very interesting application is the detection and segmentation of cell nuclei either in a complete organism or only in a part of interest such as an organ. In addition, we want the process to be fully automated.

In this section, we propose an approach that addresses each of these goals and we illustrate it by applying the method to our system model of interest, namely the *Drosophila* wing. As stated earlier, we consider that the *Drosophila* wing is a model uniquely suited for a systems biology approach and for studying the genetic program that governs the growth and shape of an organ[45,48]. As a reminder, the wing disc includes a region called the wing pouch (**Section 1.3**). In the adult wing, the wing pouch gives rise to the wing blade while the part surrounding it (called hinge) forms a flexible link attaching the wing blade to the body wall of the fly (**Section 1.3**). Many research projects focus on the wing pouch system to study the role of long-range morphogen in tissue patterning[149] and more generally in wing development[150,151]. Therefore, we propose to integrate the wing pouch model inferred in an unsupervised way in **Section 4.2** and the automatic cell nuclei detection method described below to segment and

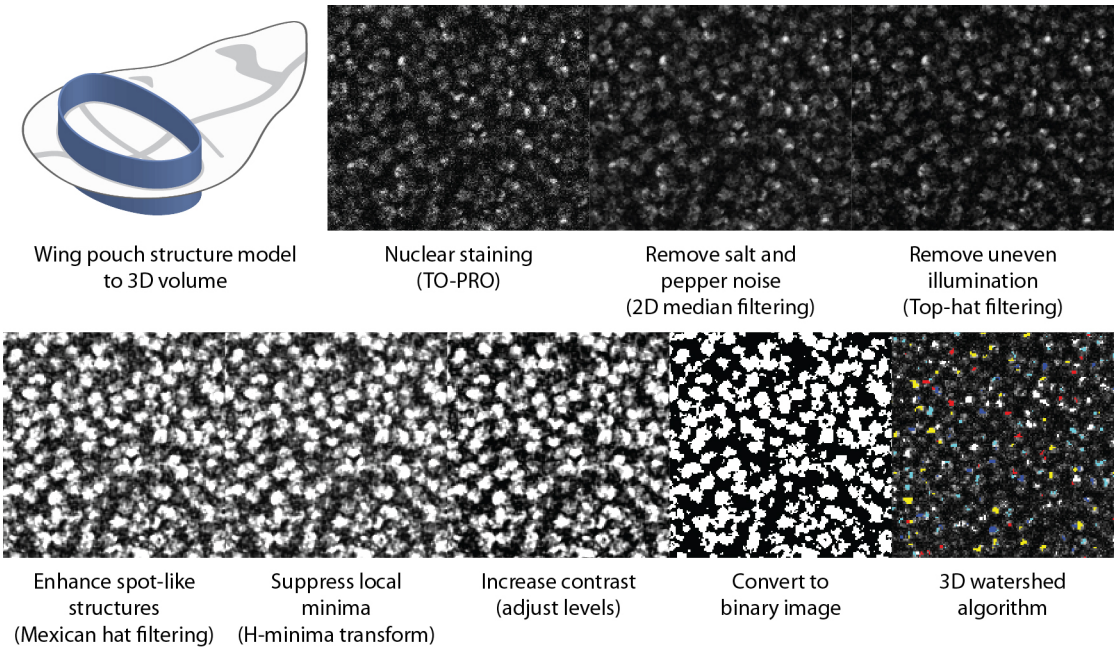


Figure 4.29: Unsupervised 3D cell nuclei detection in the *Drosophila* wing pouch. The parametric structure model of the wing pouch previously identified (**Section 4.2**) is derived to define a subspace or *area of interest* inside the wing imaginal disc. The 3D volume inside which cell nuclei will be segmented is achieved by extruding the 2D wing pouch model along the z-axis of the image space. Here, cell nuclei were stained with TO-PRO in in a stack of confocal fluorescence images (**Appendix D.1**). Several filters are then applied to increase the ratio signal/noise before obtaining a binary representation of the nuclei. Finally, the watershed transform[152] provided by Matlab is applied to segment the cell nuclei. The last image shows a section of the 3D image volume where individual nuclei are labelled using different colors. See **Video S7** for a 3D rendering of output of our cell nuclei detection method.

count the number of nuclei solely in the wing pouch. This same approach can be extended to any subsystems or organs of a given organism.

So far, we have considered the wing pouch as a two-dimensional system because the wing imaginal disc is single-layered and flat in the region of the pouch (**Fig. 1.1**). This single-layered sheet is actually composed of columnar cells whose nuclei are at different z locations (**Fig. 1.1B**). Therefore, we extrude the pouch structure model along the z-axis to define a 3D volume that includes entirely the target columnar cells (**Fig. 4.29**). Note that more complex system morphologies identified by their respective structure detection method would be returned directly as 3D models. We then use the obtained 3D shape to defines a volume of interest (VOI) to which the cell nuclei detection is constrained.

In our experiments, the cell nuclei were stained with TO-PRO-3 which is a fluorescent dye we use to make the cell nuclei visible in the wing disc. Note that we could also have used histone GFP constructs[153]. Another approach is to label cell membranes before applying a cell

boundary detection method. However, cell membranes are sometimes not already formed in early stages of development as it is the case of the *Drosophila* embryo[154]. For the *Drosophila* embryo, we recommend to use DAPI-like labellings[155].

A review of the literature at the beginning of our project revealed that there were not so many automatic procedures for cell nuclei detection in 3D microscopy images. Among them, only a few make their implementation available to the community[156,157].[b] Therefore, we decided to develop one and to make it available as part of our open-source image processing toolkit. The method we implemented is based on a *watershed transform*[152] provided by the Image Processing Toolbox of Matlab. Prior filtering of the confocal fluorescence images is required to remove noise or local irregularities, which usually lead watershed algorithms to important over-segmentation. The principle of watershed algorithms is to flood a 3D space (or a 2D image seen as a topographic relief) from its local minima before building barriers when different sources are meeting[120,152]. The barriers are then used to distinguish the volume of individual cell nuclei.

Common parameters to existing cell nuclei detection methods are an estimation of the diameter of a cell nuclei and an estimation of the minimum distance between nuclei[157,158]. Here, we used these parameters to define several filters that we apply to increase the signal-to-noise ratio in order to increase the accuracy of the watershed transform (**Fig. 4.29**). To assess the performance of the selected parameters, we manually defined a volume in a stack of confocal fluorescence images inside which we manually labeled cell nuclei. We repeated this for wild type and *pent*$^{2\text{-}5}$ mutant wings and for different time points between 53 and 114 hours AEL. Typically, each rectangular volume was defined to cover and fall entirely in one half of the wing pouch model (the anterior compartment to be precise). We the used the data from the manual counting to fine-tune the parameters of the cell nuclei detection method.

As a suggestion for improvement, we propose to apply an evolutionary algorithm for optimizing the parameters of the nuclei detection method. Evolutionary algorithms (EA)[159,160] and particle swarm optimization (PSO)[161,162] are both population-based heuristic optimization techniques that try to iteratively generate better candidate solutions with respect to a quantitative description of quality, also called fitness function[93,160]. In our application, the fitness function would be a measure of the distance between the automatic and manual cell nuclei counting. Evolutionary algorithms spend most of the time evaluating the candidate solutions (so-called *individuals*), which corresponds here to run the cell nuclei detection method for every system sample that composes the benchmark. Of course, a set of optimized parameters depends on the intrinsic information of the input images, that is, using a different techniques to mark the cell nuclei would require to run once again the optimization process. The maximum runtime of the optimization procedure such as a generational genetic algorithm[159,160] is given by the maximum number of generations × the number of individual evaluated per generation × the time spent evaluating the fitness function of the new individuals. If a set of

[b] Some commercial solutions exist, however, the details of their implementation are not published.

parameters for the cell nuclei detection method is initially available and has been shown to achieve relatively good segmentation, a CMA-ES[163,164] or PSO[165] optimization algorithm can be applied to find parameter values that lead to even better performance.

It is important to note that there are no segmentation techniques that allow to perfectly identifying every single cell nucleus. This is true at least for the *Drosophila* wing stacks of confocal fluorescent images stained with TO-PRO. We also imaged a few wings using an histone GFP construct[166] and we observed that the output is very similar to those obtained with TO-PRO. The variability in the marker intensity and the diffusion of the fluorescence are the principal difficulties to overcome when developing an automatic segmentation algorithm. Even for the human eye it is sometimes difficult to clearly distinguish between two cell nuclei that are particularly close. However, even if the segmentation method applied to identify cell nuclei does not lead to a flawless counting, the data collected are still useful to perform quantitative comparative analyses, for example when considering the relative number of cells in wild type and mutant *Drosophila* wings and for wings imaged at different time points.

Finally, we report later in **Figure 4.43** the evolution of the number of cell nuclei in both wild type and *pent*$^{2\text{-}5}$ deficient *Drosophila* wings imaged between 70 and 110 hours AEL.

4.6 *Drosophila* wing pouch model repository

Figure 4.30 shows two quantitative descriptions of the wild type and *pent*$^{2\text{-}5}$ mutant *Drosophila* wing imaged at 80 hours after egg laying. We achieve this result using the unsupervised detection and modeling of the *Drosophila* wing pouch structure (**Section 4.2**), the quantification of gene and protein expression in two and possibly three spatial dimensions (**Section 4.3**), and the method that we developed to integrate structure and expression data into a single and robust quantitative description of the biological system considered (**Section 4.4**).

We have then organized the models of the *Drosophila* wing before making them available with a web application (tschaffter.ch/projects/wingj) to visualize these descriptions and thus observe the difference in term of morphology and gene expression level between wild type and *pent*$^{2\text{-}5}$ mutants and between wings imaged at different stages of development. The set of models currently available has been generated from any possible combinations of the following parameters:

- **Genotypes:** wild-type and *pent*$^{2\text{-}5}$ mutant wings
- **Time points:** 80, 90, 100, and 110 hours AEL
- **Fluorescent proteins:** Brk-AB, Dad-GFP, Omb-AB, Pmad-AB, and Sal-AB

For each combination of parameters, we have quantified fifteen to thirty wings to generate robust structure models. The dashed line in **Figure 4.30** provides information about the

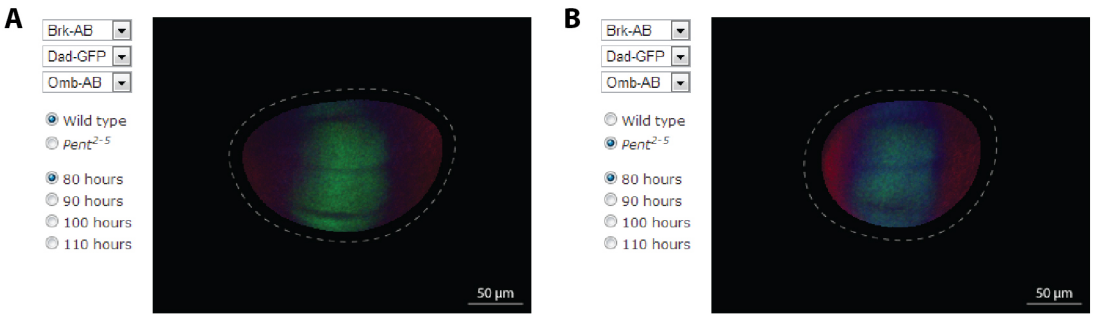


Figure 4.30: Repository of *Drosophila* wing pouch models. The methods we developed for unsupervised detection and modeling of the *Drosophila* wing pouch structure (**Section 4.2**), quantification of gene and protein expression in two and possibly three spatial dimensions (**Section 4.3**), and automatic detection and segmentation of cell nuclei solely inside the *Drosophila* wing pouch (**Section 4.5**) enable the automatic generation of multiscale and robust quantitative descriptions of the *Drosophila* wing. A web application is available on the project website[c] to visualize these descriptions and thus observe the difference in term of morphology and gene expression level between wild type and $pent^{2\text{-}5}$ mutants and between wings imaged at different stages of development. The expression of up to three gene products can be shown at the same time (in red, green, and blue). Here, each structure model have been generated from fifteen to thirty wings. The dashed line represents the standard deviation of the structure. Each expression map (Brk, Dad, Omb, Pmad, and Sal) has been averaged from five to ten wings. We show in panels **(A)** and **(B)** the difference in morphology and expression domains between wild type and $pent^{2\text{-}5}$ mutant 80-hour-old *Drosophila* wings. The representations are always given in the canonical orientation (anterior is left, dorsal is top).

standard deviation of the shape of the wing pouch. Moreover, five to ten wings have been used to compute each expression map. The web interface allows to display up to three gene products at the same time (in red, green, and blue). **Figure 4.30B** shows the effect of the $pent^{2\text{-}5}$ mutation on the morphology and expression domains of the *Drosophila* wing. Note that the wing models generated by WingJ are always given in their canonical orientation (anterior is left, dorsal is top).

We hope to further extend this collection of robust quantitative descriptions with additional mutations, time points, and fluorescent proteins, but also to include a large variety of biological systems with the help of the community. We expect that this will become easier as the number of unsupervised and weakly supervised detection methods implemented in WingJ will increase.

4.7 Inference of the *Drosophila* wing developmental network

Researchers have proposed a plethora of methods for reverse engineering the complex network of interactions between the genes and their RNA and protein products (also called *regulatory program*) from spatial and temporal high-throughput gene expression data[51,167] (see **Chap-**

ter 2). Regulatory networks are often represented as directed, signed graphs in which nodes typically represent genes or transcription factors (TF). In this context, edges correspond to enhancing or inhibitory regulations that affect gene transcription rates. Network inference methods rely on various computational approaches such as correlation[56], mutual information (MI)[57,58], ordinary differential equations (ODE) models[60,168], Bayesian networks[169], hybrid algorithms[71], or based on the *wisdom of the crowd*[4].

Reverse engineering methods for regulatory structure and dynamics inference of gene networks from non-spatial steady state and time series expression data have been successfully applied in various biological systems (*E. coli*[6], *S. cerevisiae*[5], *S. aureus*[170],etc.). However, a challenging issue remains the reconstruction of the structure and non-linear dynamics of developmental gene networks from spatial and temporal data. A classical example is the inference of the gap gene network, which is involved in segment determination during early development of *Drosophila melanogaster*[40,148]. The proposed algorithms take as input time series data (1D expression profiles measured along the D/V compartment boundary of the embryo and at different time points) and possibly incorporate prior knowledge to infer the interactions between the morphogens Bicoid (Bcd), Caudal (Cad), and Toll (Tll), and the gap gene proteins Hunchback (Hb), Krüppel (Kr), Knirps (Kni), and Giant (Gt). In addition to the higher complexity of the models that have to be apply, a major problem is the generation of spatial and temporal quantitative datasets (sample collection, immunostainings, and image acquisition) that require significant amount of time and effort. Note that this is actually where lies most of our motivation for developing methods that enable the automatic quantification of biological systems at different scales. The generation of robust, multiscale quantitative descriptions of these biological systems achieved in **Section 4.4** should then provide valuable quantitative datasets for reverse engineering developmental gene regulatory networks.

Here, we consider expression datasets having more than one spatial dimension and propose a relatively simple method to infer the structure of developmental networks. We then use if for the reconstruction of a six-gene network involved in the development of the *Drosophila* wing. First, measurements of gene and protein expression levels must be taken in the biological system of interest. The *Drosophila* embryo is one of the most comprehensively investigated system model because its anterior-posterior (A-P) and dorsal-ventral (D-V) patterning systems are largely independent in the trunk region of the embryo[40]. Thus, the patterning of the embryo along the A-P axis provides a convenient one-dimensional system for developing developmental network inference methods. However, extending expression datasets from one to two spatial dimensions is not trivial. The main issue is the collection and representation to give to individual measurements to enable their integration and thus obtain reliable data that we can use for network inference. For one-dimensional datasets, a simple approach like the one we describe in **Section 4.4.2** can be applied. Starting with two dimensional datasets (for now we only consider spatial dimensions), the variability in the system morphology hinders the definition of a representation enabling the integration of individual expression measurements. In **Section 4.3**, we formally describe a method that addresses directly this issue and allows us to generate maps describing gene expression in two spatial dimensions.

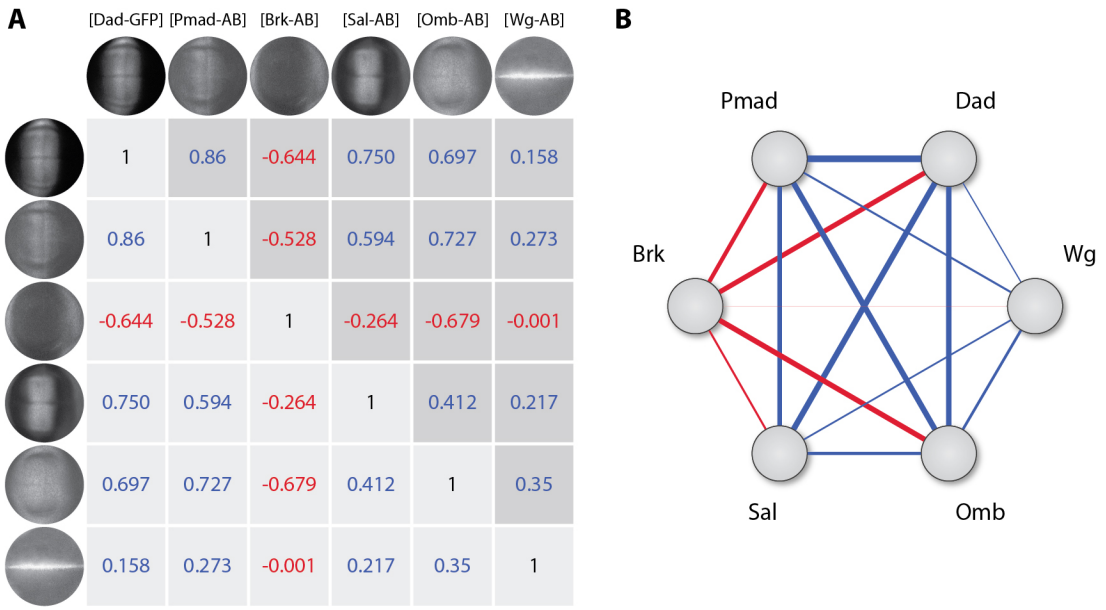


Figure 4.31: Inference of a six-gene *Drosophila* wing developmental network. **(A)** Table of correlation coefficients r between any pair of mean expression maps. Individual expression maps are generated from the structure and expression information quantified in *Drosophila* wings (**Sections 4.2** and **4.3**). Mean expression maps are obtained by averaging many individual maps thanks to the convenient circular representation we gave them. Here, the expression maps report the domain of expression of six gene products, namely Dad, Pmad, Brk, Sal, Omb, and Wg in 90-hour-old wild type wings. Positive and negative values are given in blue and red respectively. **(B)** Topology of the six-gene network inferred from 2D (possibly 3D) spatial snapshots of gene and protein expression data. The regulatory network is represented as an undirected, signed graphs in which nodes represent genes. Blue and red edges correspond to enhancing ($r > 0$) and inhibitory ($r < 0$) regulations that affect gene transcription rates. The width of the edges is proportional to the absolute value of r and can be interpreted as the probability to have a direct interaction between two gene products (but not as the *affinity* between the protein of a gene A and the promoter of a gene B).

Note that the proposed model can then be fairly easily extended to three spatial dimensions; the tricky part being the extension from one to two spatial dimension. As a remainder, we use the parametric model of the system morphology inferred in **Section 4.2**, which is composed of a B-spline describing the outer boundary of the system and two A/P and D/V compartment boundaries, to define a grid used to define where expression measurements should be taken. We then organize the measurements while normalizing the spatial dimensions to generate disc-shapes expression maps. Finally, individual expression maps belonging to the same class of experiments are integrated to obtained mean expression maps (**Fig. 4.30**), therefore providing robust and valuable two-dimensional (ultimately three-dimensional) *snapshots* of the domains of gene and protein expression required for accurate reconstruction of developmental gene networks in *3D* biological systems (**Section 4.4.3**).

We propose the following method for the first network inference using 2D spatial expression maps. To obtained a quick insight into the structure of a gene regulatory network involved in the development of the *Drosophila* wing, we compute the *2D correlation coefficient r* given in (4.29) between any pair of mean expression maps available for Dad, Pmad, Brk, Sal, Omb, and Wg from 90-hour-old wings.

$$r = \frac{\sum_m \sum_n \left(A_{mn} - \overline{A}\right)\left(B_{mn} - \overline{B}\right)}{\sqrt{\left(\sum_m \sum_n \left(A_{mn} - \overline{A}\right)^2\right)\left(\sum_m \sum_n \left(B_{mn} - \overline{B}\right)^2\right)}} \tag{4.29}$$

A and B are two individual expression maps similar to the one shown in **Figure 4.27B**. In this context, an expression map is a single square image. The expression data themselves are included in a disc centered on the square image and whose diameter is the width of the image. The remaining pixels of the image are set to black (A_{mn}=0). A_{mn} represents the pixel value in the first expression map at the pixel location (m,n) (we only consider pixels that fall inside the disc of expression) and $\overline{A}$ is the average of all the pixel values considered.

We use the correlation coefficient r to measure of the strength of linear dependence between the domain of expression of two proteins. Note that this relation may not be dependent in practice. Note that there are many different approaches that would be more suitable for network inference (see short review in the second paragraph of this section). However, measuring correlation allows to quickly get insight in the structure of the target network. This first detection may even benefit to more complex algorithms applied successively to refine the structure or generate dynamical models. The correlation coefficient r takes values between -1 ($r(A,-A)$ where $-A$ is the negative image associated to A) and 1 ($r(A,A)$). An image correlated with a rotated version of itself usually results in $r < 1$. **Figure 4.31A** reports the correlation coefficient r between any pair of proteins constituted from Dad, Pmad, Brk, Sal, Omb, and Wg. Positive and negative r are given in blue and red, respectively. The expression maps are already optimally aligned because we used the coordinate system described in **Section 4.3.3**, which is derived from the inferred model of the morphology or structure of the *Drosophila* wing pouch.

Figure 4.31B shows an illustration of the undirected, signed network structure inferred based on the correlation between 2D expression maps. The nodes represent genes. Blue and red edges correspond to enhancing ($r > 0$) and inhibitory ($r < 0$) regulatory interactions that affect gene transcription rates. The width of the edges is proportional to the absolute values of r given in **Figure 4.31A**. It is important to note that r can not be interpreted as the *weight* of an interaction, which traditionally represents the affinity between the protein of a gene A and the promoter of a gene B[56,60]. However, a high absolute correlation coefficient value $|r|$ indicates that it is likely to have a direct interaction between two genes. This information can then be validated in biological experiments using knockout and/or knockdown perturbations[51]. Correlation alone is usually not sufficient to identify indirect interactions. Moreover, the inference

of *directed* structures of gene networks can be achieved from temporal expression data or times series and then look at the causal relationships between two genes, and perturbation experiments (e.g. knockout experiments). Here, we only have four time points (80, 90, 100, and 100 hours after egg laying) but only data from 90-hour-old *Drosophila* wings are displayed in **Figure 4.31**.

The method applied succeeds in providing some insight into structure of the six-gene network. The advantage of network inference methods that relies on correlation[56] or mutual information (MI)[57,58], for instance, is that they can be applied to networks including thousands of gene and transcription factors. In addition, algorithms have been proposed to reverse engineering dynamical models of these networks using ordinary differential equations (ODE) or Bayesian networks[60,169]. However, the algorithms that propose to infer details dynamical models can not be applied to genome-wide networks. That's why relatively simple methods such as correlation- or MI-based are still in hybrid inference methods to provide a first structure of the regulatory interactions before endowed with them dynamical models of gene regulation accounting for both transcription and translation, for instance.

We observe that the six-gene *Drosophila* wing network is fully connected. This was expected for mainly two reasons. The first is that any correlation coefficient is different from zero (**Fig. 4.31A**), which then lead to the creation of an edge in **Figure 4.31B**. Assumptions can be made to define a threshold to apply on r in order to discard edges corresponding to small values and so obtain a sparser network structure. Moreover, the generation of a list of predictions sorted in descending order of $|r|$ (see below) can be directly used to choose the first interactions to assess *in vivo* experiments. For example:

```
G861   G496   0.987
G225   G931   -0.856
G813   G477   -0.844
G624   G850   0.816
...
```

where each line defines an interaction oriented from the first gene to the second gene. The third element is the correlation coefficient r. Typically, prediction files include in one of their columns a metric which quantifies the degree of belief that a given interaction is effectively present in the network to reverse engineer. This *confidence level* is then used to sort the list of predictions in descending order so that the interactions that are the most likely to be present in the target network are at the top of the file[51].

The edges between Wg and the other nodes of the six-gene network would not be considered in a first time for *in vivo* validation based on the result shown in **Figure 4.31B**, therefore saving time and resources for studying the remaining interactions. Finally, the interactions between Dad, Pmad, Brk, Sal, and Omb have already been extensively studied and it has been shown

that these genes are working tightly together to ensure the growth and proper patterning of the *Drosophila* wing[45,46].

4.8 Quantitive description of the developing type*Drosophila* wing

We present below a collection of datasets related to the developing *Drosophila* wing. From stack of confocal fluorescence image, we have applied the detection and segmentation method described in **Section 4.2** to generate a parametric model that describes the morphology of the wing pouch. This structure includes the A/P and D/V compartment boundaries, and the outer boundary of the pouch. We have then used this model to quantify gene and protein expression in a systematic way. Moreover, these heterogeneous datasets have being integrated to generate robust and reliable quantitative description of the *Drosophila* wing pouch (**Fig. 4.42**). It is important to note that data collection and integration has been performed using WingJ, the image processing application developed in the frame of this thesis.

The first datasets provide information about the morphology of the wing at different time points during development (**Figs 4.32** to **4.36**). Expression profiles (**Fig. 4.39**) and maps (**Figs 4.40** and **4.41**) are also shown for different genes involved in the development of the wing. We show that the above method can also be used to quantify the effect of a mutation. Here, we consider *pent* deficient wings. The $pent^{2\text{-}5}$ mutation has been shown to play a role in scaling the gradient activity of the morphogen Dpp, which in turn inhibits the growth of *Drosophila* wings[50]. **Figure 4.42** illustrates well the quantification of the effect of this mutation on the domain of expression of several genes and the resulting shape of the wing. Finally, **Figure 4.43** reports the number of nuclei detection in wild type and $pent^{2\text{-}5}$ mutant wings at different time points. For the sake of simplicity, observations are directly included in the captions of the following figures.

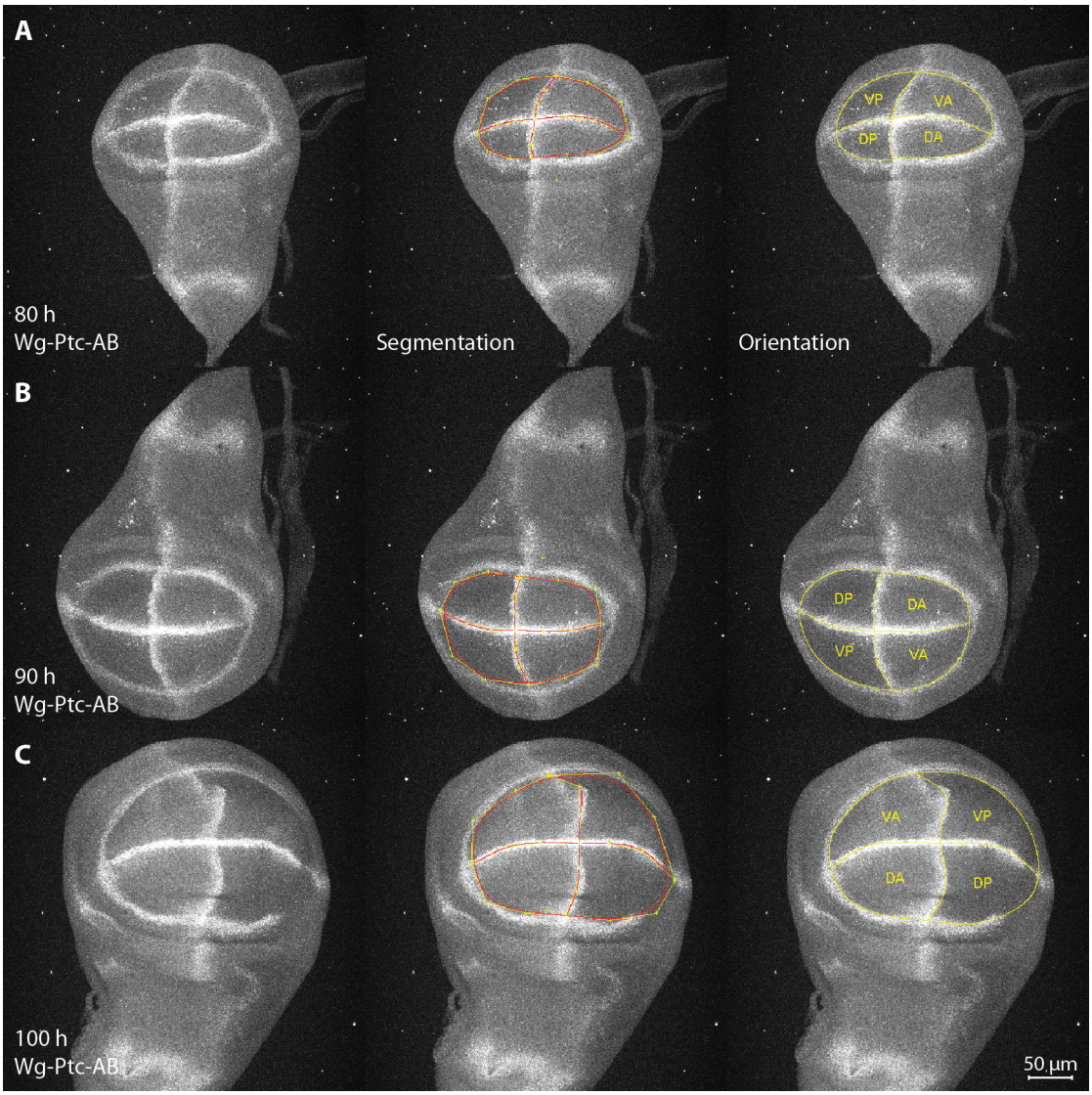


Figure 4.32: Unsupervised detection and segmentation of the *Drosophila* wing pouch. The expression of Wg-Ptc is used to label the structure of the wing pouch imaged at **(A)** 80, **(B)** 90, and **(C)** 100 hours after egg laying (AEL). The second column shows the output of the unsupervised detection and segmentation algorithm available in WingJ. The output is a parametric model that provides a quantitative description of the morphology of the pouch. Users can interact with this model to change its shape. This is achieved intuitively by moving control points ('+' vertices placed around the structure). After manual fine-tuning, an algorithm is applied to automatically identify the A/P and D/V orientation of the structure model.

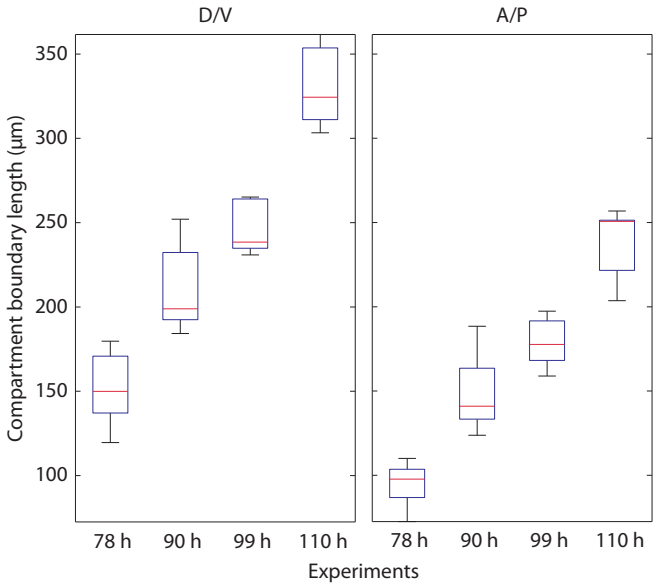


Figure 4.33: Length of the A/P and D/V boundaries in wild type *Drosophila* wings. As expected, the D/V compartment boundary is systematically longer than the A/P boundary. At late third instar, the wing pouch starts to evert in the ventral direction. This is described by the larger increase in the length of the D/V boundary between 99 and 110 hours. Between fifteen and thirty wings were used for each class of experiments.

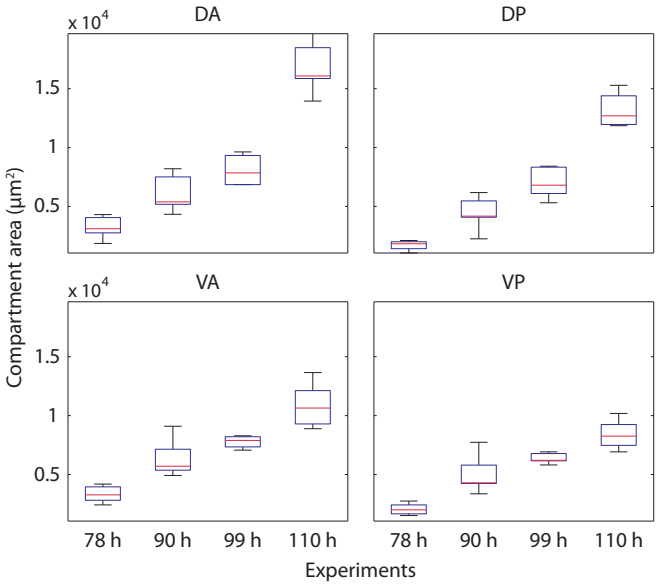


Figure 4.34: Area of the compartments DA, DP, VA, and VP in wild type *Drosophila* wings. In addition to the data shown in **Figure 4.33**, we observe that the eversion of the wing pouch, which starts around 110 hours AEL, leads to a large increase in the area of the dorsal compartment (compartments DA and DP). This is consistent with previous observations[43,47] reporting that the wing pouch everts towards the ventral direction. Between fifteen and thirty wings were used for each class of experiments.

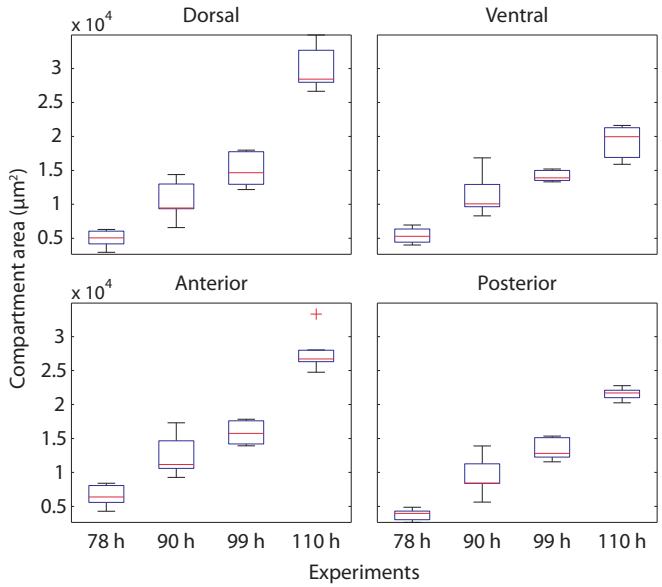


Figure 4.35: Area of the compartments D, V, A, and P in wild type *Drosophila* wings. The dorsal compartment D is composed of the two compartments DA and VA, for example (**Fig. 4.32**). Here, we observe clearly that the dorsal compartment starts to grow significantly faster than the ventral compartment after 100 hours AEL (Mann-Whitney U-test, $P < 0.01$), which corresponds to the beginning of the wing eversion process[43,47]. Between fifteen and thirty wings were used for each class of experiments.

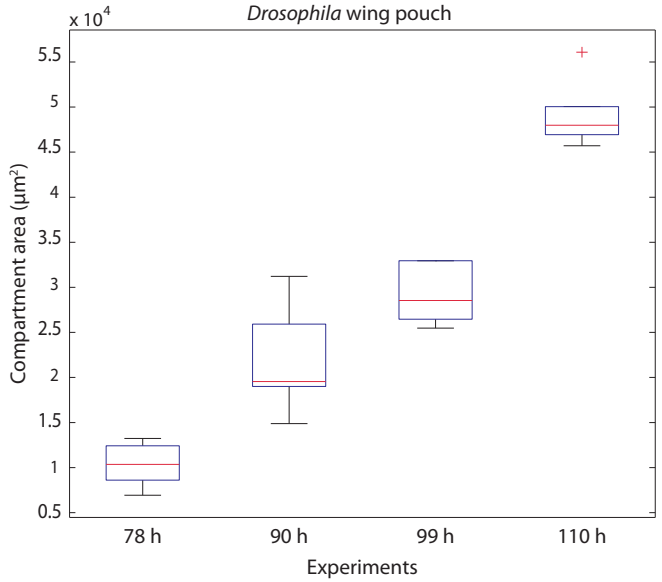


Figure 4.36: Area of the wing pouch in wild type *Drosophila* wings. Here, we report the evolution of the area of the entire wing pouch between 78 and 110 hours. The wing pouch includes the four compartments DA, DP, VA, and VP whose areas are reported in **Figure 4.35**. The eversion process, which starts around 110 hours AEL, clearly affects the overall growth of the pouch as the difference in size between 99 and 110 hours is significant (Mann-Whitney U-test, $P < 0.01$). Between fifteen and thirty wings were used for each class of experiments.

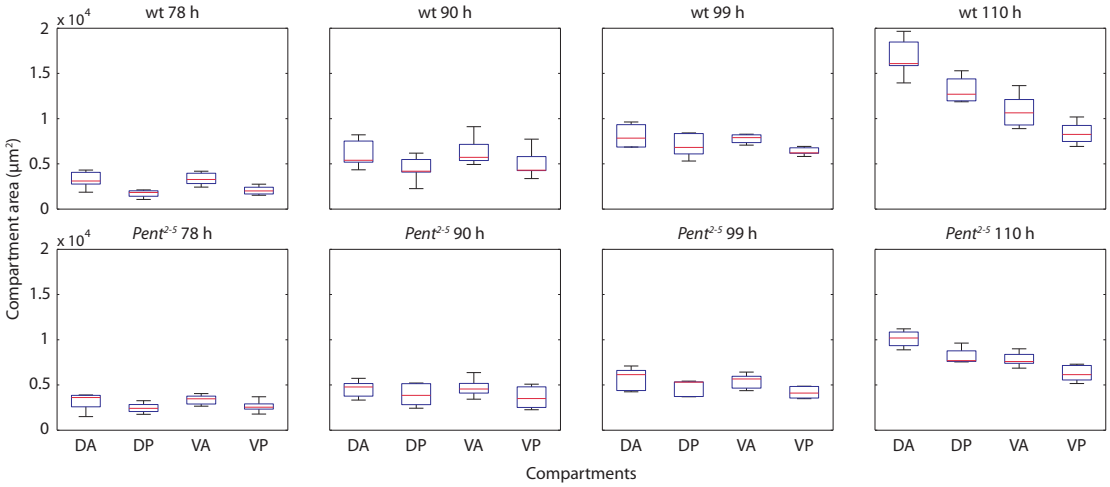


Figure 4.37: Area of the compartments DA, DP, VA, and VP in wild type and *pent*$^{2\text{-}5}$ mutant *Drosophila* wings. The first line reports the areas of wild type wing pouch imaged between 78 and 110 hours AEL. At 110 hours, the dorsal compartment (DA+DP) is significantly larger than the anterior compartment (Mann-Whitney U-test, $P < 0.01$) which shows clearly that the eversion of the pouch follows the dorsal-ventral axis instead of the anterior-posterior axis. The area of each compartment is significantly smaller in 110-hour-old *pent* deficient wings than in corresponding wild type wings (Mann-Whitney U-test, $P < 0.01$).

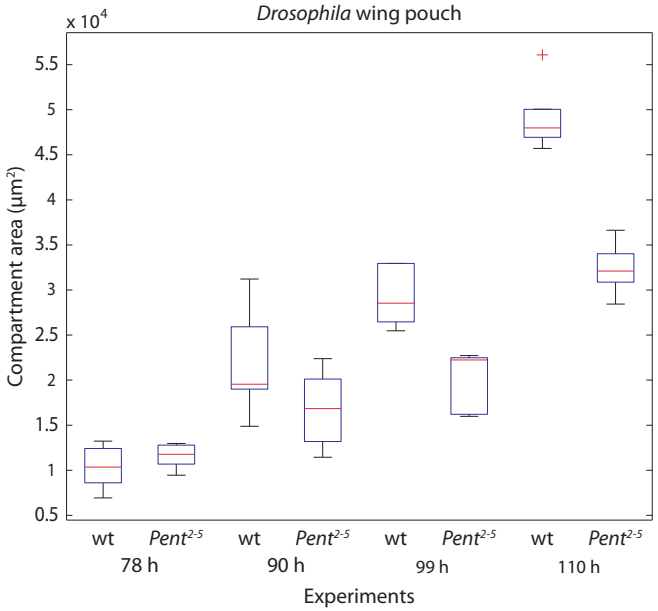


Figure 4.38: Area of the wing pouch in wild type and *pent*$^{2\text{-}5}$ mutant *Drosophila* wings. Here, we report the evolution of the area of the entire wing pouch between 78 and 110 hours in both wild type and *pent*$^{2\text{-}5}$ mutant wings. The *pent*$^{2\text{-}5}$ mutation has been shown to play a role in scaling the gradient activity of the morphogen Dpp, which in turn inhibits the growth of *Drosophila* wings[50]. It appears here that the difference in area is not significantly different in 78- and 90-hour-old wings (Mann-Whitney U-test, $P > 0.05$). However, this difference becomes clearly significant in 99- and 110-hour-old wings (Mann-Whitney U-test, $P < 0.01$).

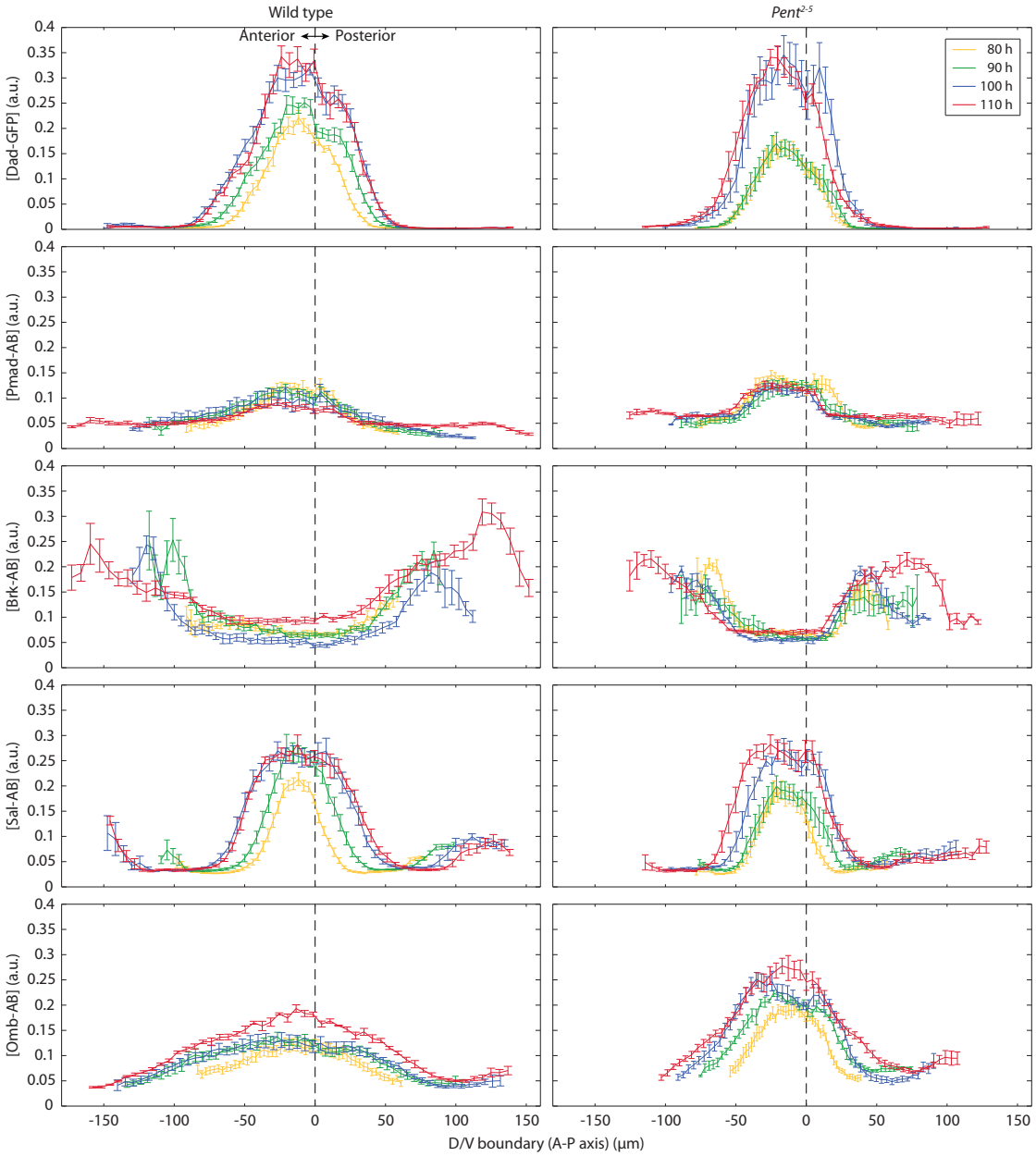


Figure 4.39: Expression profiles of Dad, Pmad, Brk, Sal, and Omb in wild type and *pent*$^{2-5}$ mutant wings. We have measured the expression profiles along the D/V boundary (A-P axis). Expression values initially range from 0 (gene is not expressed) to 255 (gene is fully expressed) and are given in arbitrary units, which we then normalize (division by 255). The thick lines represent the mean expression and the bars the standard error. The expression profiles reported on the right show how the *pent*$^{2-5}$ mutation affects the activity gradient of Dad, Pmad, Brk, Sal, and Omb. We explain this by the fact that Pentagon (Pent) has been shown to play a role in scaling the Dpp gradient activity[50] and because Dpp down-regulates directly or indirectly the production of each of the other gene products. Between five and ten wings were used for each class of experiments.

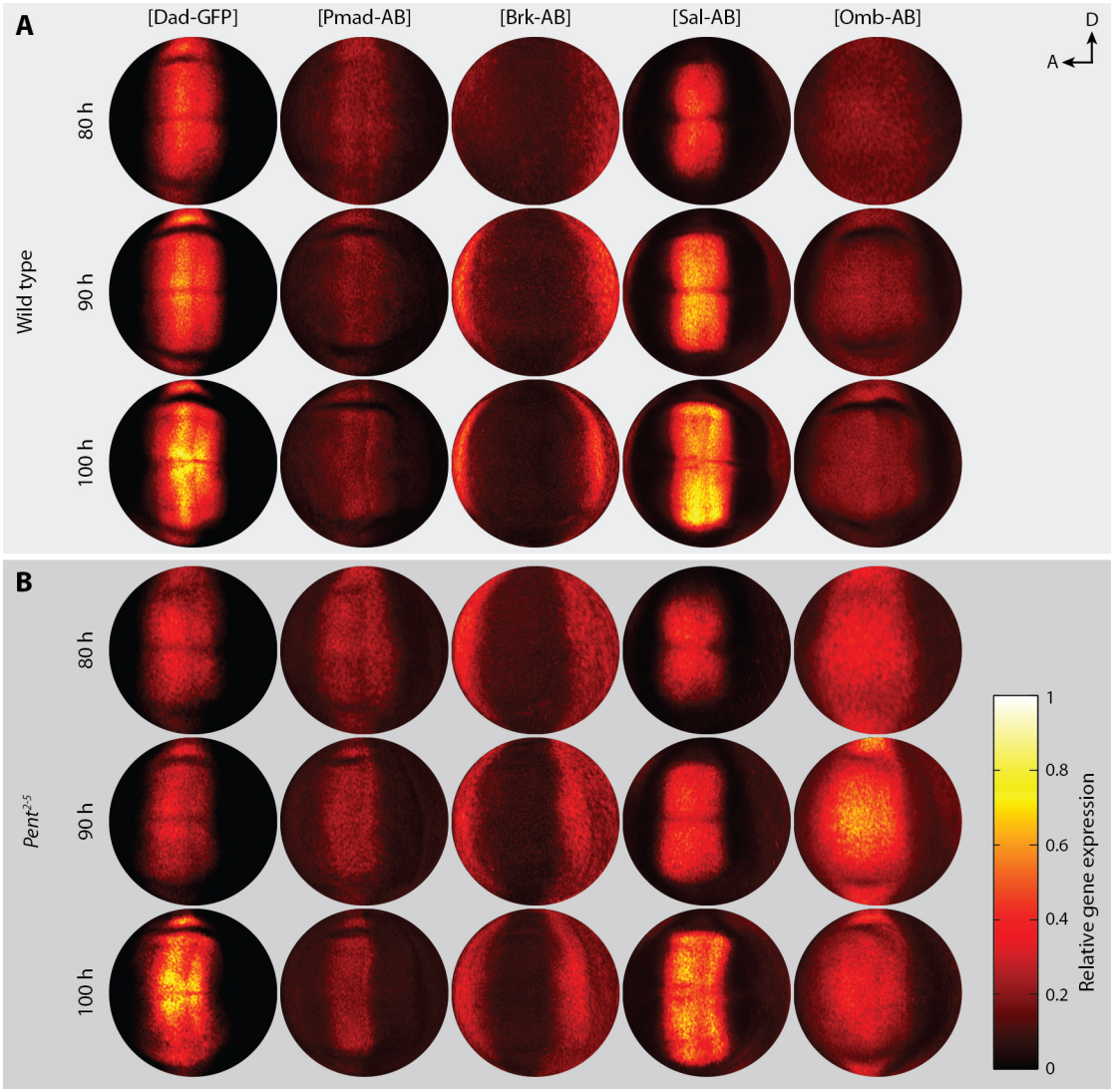


Figure 4.40: Mean expression maps of Dad, Pmad, Brk, Sal, and Omb in wild type and *pent*$^{2\text{-}5}$ mutant wings. **(A)** For each individual wing, we infer a parametric model of the pouch structure using the unsupervised detection method introduced in **Section 4.2**. This model defines a coordinate system (**Section 4.3.3**) used to quantify mRNA and protein expression in the space of the model, i.e. inside the wing pouch. The expression data collected from many wings are then represented as disc-shaped *expression maps* (**Section 4.3.5**) before integrating them into a reliable, mean expression map (**Section 4.4.3**). Expression values initially range from 0 (black, gene is not expressed) to 255 (white, gene is fully expressed) and are given in arbitrary units, which we then normalize (division by 255). **(B)** We see in this panel that the *pent*$^{2\text{-}5}$ mutation affects the activity gradient of Dad, Pmad, Brk, Sal, and Omb. As mentioned previously, it has been demonstrated that Pentagon (Pent) plays a role in scaling the Dpp gradient activity[50] and because Dpp down-regulates directly or indirectly the production of each of the other gene products. For instance, the activity gradient of Pmad, which is a readout of Dpp signalling, forms more compact stripes of Pmad expression along the A/P boundary in *pent* deficient than in wild type wings. Between five and ten wings were used for generating each expression map.

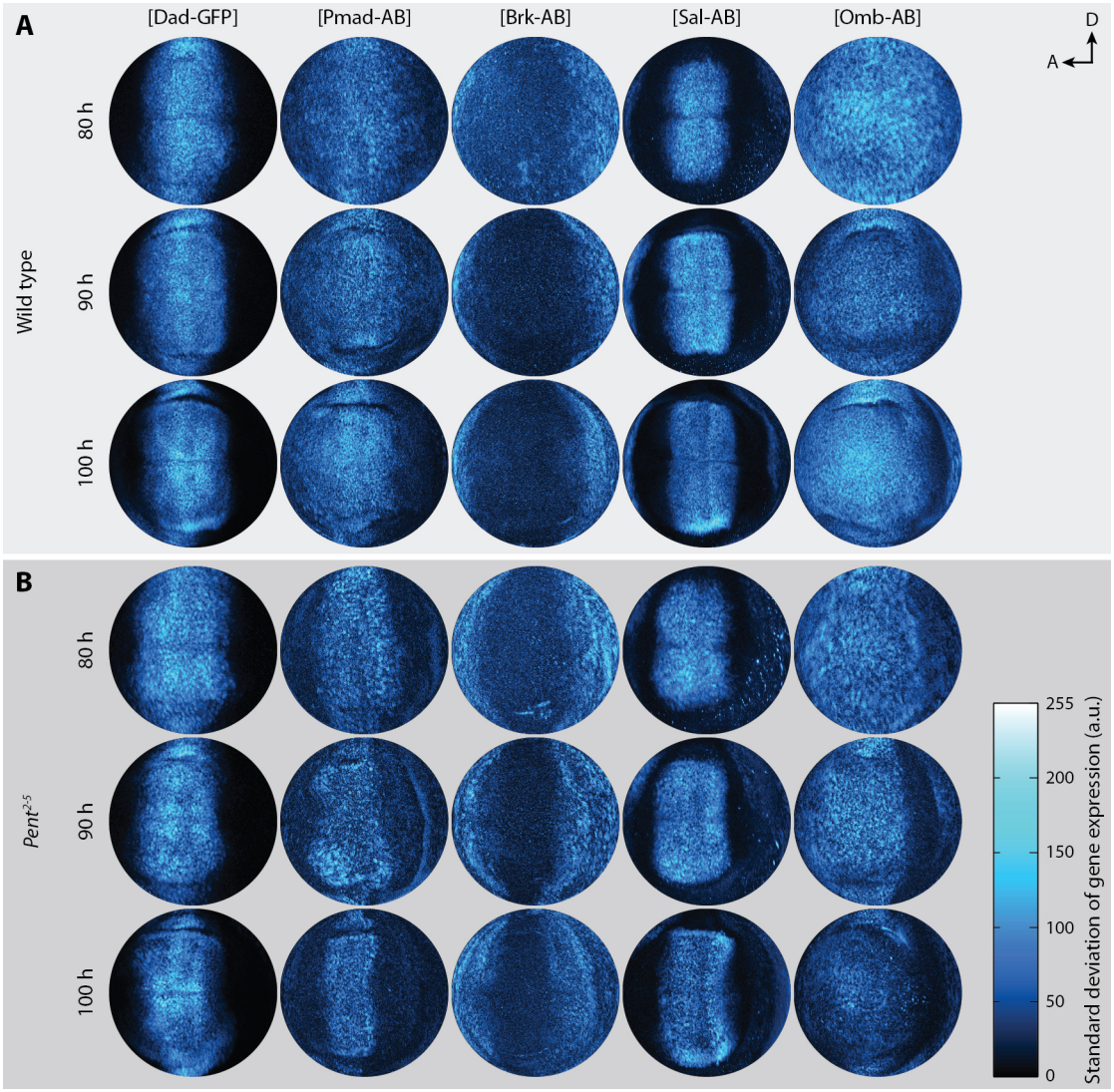


Figure 4.41: Standard deviation expression maps of Dad, Pmad, Brk, Sal, and Omb in wild type and *pent*$^{2-5}$ mutant wings. (A-B) These expression maps were generated along with the mean expression maps shown in **Figure 4.40**. Reporting the standard deviation provides an additional tool to ensure that the generation of structure models is consistent across many experiments (wings). If the structure model inferred for a wing differs from other experiments, the expression of its gene products would be sampled at different locations and thus regions with large values of standard deviation would appear on the standard deviation expression map. If this happens, the collected data can be imported back into WingJ and maybe modify the shape of the structure model so that it describes more accurately the morphology of the system. Standard expression values are reported in absolute and arbitrary units ranging from 0 (black) to 255 (white). Note that these expression map look very rugged because we used nuclear markers. Between five and ten wings were used for generating each expression map.

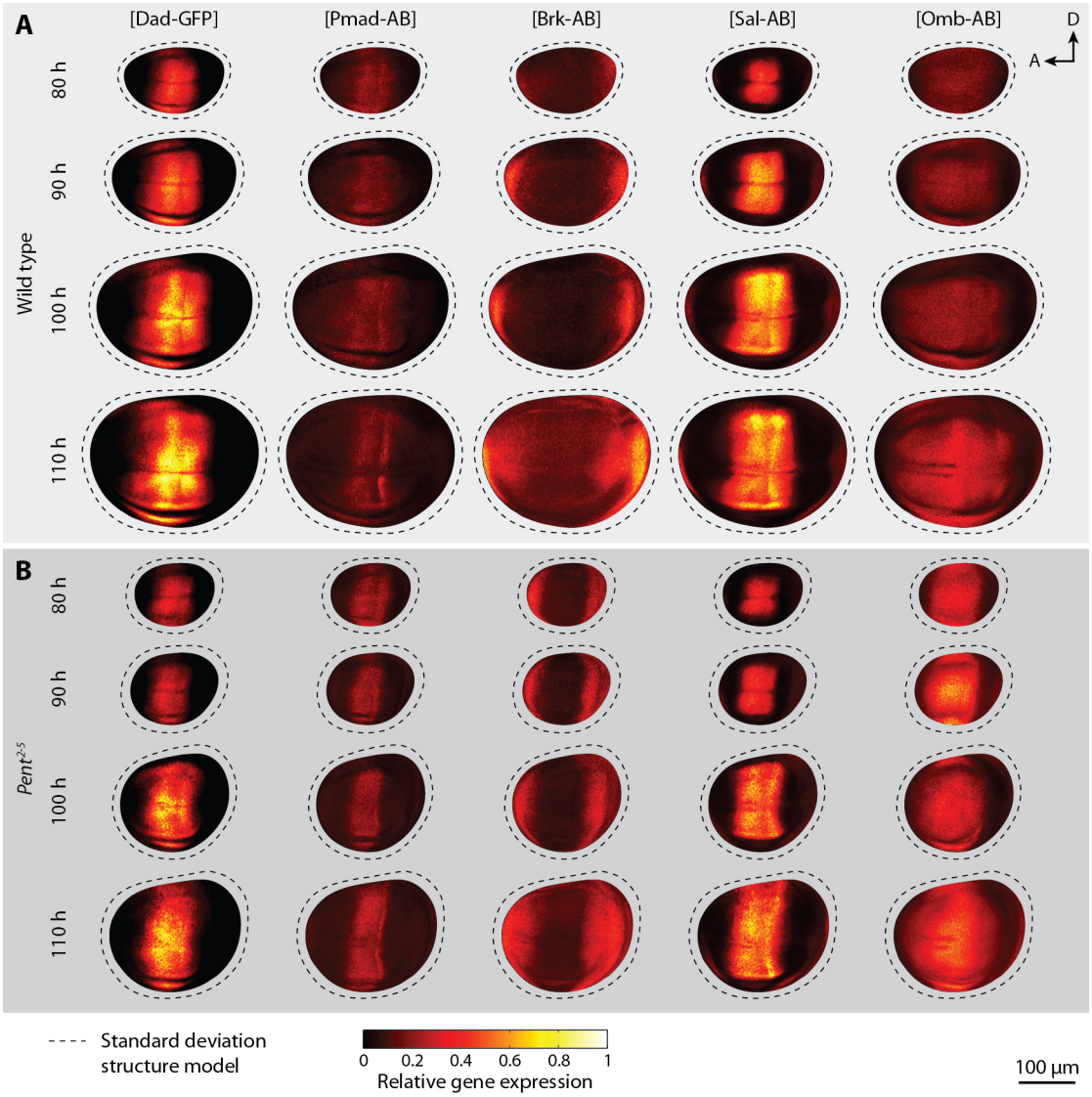


Figure 4.42: Robust and reliable quantitative descriptions of the *Drosophila* wing pouch. **(A)** The mean expression maps shown in **Figure 4.40** are combined with the mean structure model generated for each type of wings in order to provide a robust and reliable description of the developing wing. Here, each mean structure model has been computed by integrating the individual structure model of 15-30 wings. The individual models have been obtained using the unsupervised detection and segmentation method introduced in **Section 4.2**. The dashed line represents the standard deviation of the shape of the pouch. **(B)** We can precisely observe in this panel the effect of the $pent^{2\text{-}5}$ mutation on the wing size, shape and patterning which is known to be regulated by the Dpp gradient activity, here affected by the $pent^{2\text{-}5}$ mutation (the activity gradient of Pmad provides a readout of Dpp signalling). Both individual and mean models have been generated using the unsupervised tools that are available in WingJ.

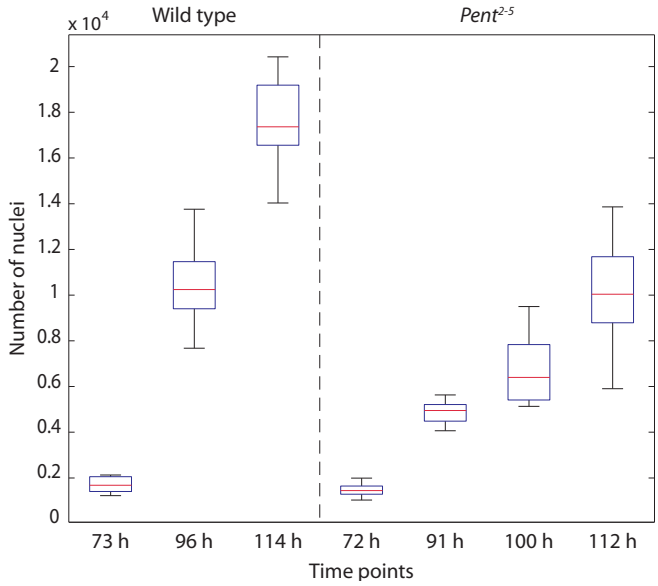


Figure 4.43: Number of cell nuclei detected in the pouch of wild type and *pent* deficient *Drosophila* wings. We apply the method described in **Section 4.2** to infer a parametric model of the morphology or structure of the wing pouch. This model is used to define a volume of interest in which we detect and segment cell nuclei in stacks of confocal fluorescence images using the approach introduced in **Section 4.5**. Here, we used the nuclear marker TO-PRO. Moreover, both structure and cell nuclei detection methods are fully automated. At 72-73 hours, the number of cell nuclei detected in the pouch is not significantly different in wild type and $pent^{2\text{-}5}$ mutant wings (Mann-Whitney U-test, $P > 0.05$). However, the difference becomes significant already at 96 hours and then at 112-114 hours (Mann-Whitney U-test, $P < 0.01$). We observe that these data are effectively and tightly related to the area of the pouch shown in **Figure 4.38**. Furthermore, the number of cell nuclei evaluated at different time points can be used to compute the cell division rates in wild type and *pent* deficient wings.

4.9 Conclusions

We propose a method for generating quantitative descriptions of the *Drosophila* wing and embryo, which are classical models for studying the genetic control of tissue size, shape and patterning. The wing pouch is the part of the wing disc that gives rise to the adult wing of the *Drosophila* (**Video S4**). A model of the morphology or structure of the wing pouch is obtained by applying a detection and segmentation method that we developed. The parametric description of its A/P and D/V compartment boundaries, and outer boundary is achieved through the design and application of multiple image processing detection modules. Each of them focus on the extraction of a specific feature of the wing structure. For this purpose, segmentation tools have been developed to enable the detection and parametric description of features such as cross-like shapes, fluorescence trajectories, and closed regions delimited by fluorescence. Also, the decomposition of the segmentation method into modules has several advantages. The performance of each detection module can be evaluated separately, thereby indicating potential ways of network reconstruction improvements. Different strategies can

also be proposed to identify the same feature of the structure, thus providing many concrete applications to novel image processing algorithms. Furthermore, the output of several modules that perform the same task could be combined using a consensus method (also called *wisdom of the crowd*) to achieve a more robust and reliable detection than simply applying the best individual method. The overall model that describes the structure of the wing pouch is then obtained by integrating the models inferred for each morphological feature. The A/P and D/V orientation of the model in the space of the image is also recovered.

The parametric model that describes the structure of each wing pouch segmented is used to define a non-orthogonal coordinate system that allows to link points that share the same relative positions in different wings. In the wing pouch, the axes of this coordinate system are defined by the A/P and D/V compartment boundaries. This coordinate system then enables the systematic quantification of gene and protein expression levels. Expression profiles that report gene and protein concentration levels along one-dimensional trajectories have first been generated. This is the most popular representation used today for the study of mechanisms involved in growth and spatial patterning of biological tissues, which are often first modelled as one dimensional systems. However, one of the main contributions of this work is the generation of two-dimensional and eventually three-dimensional spatial representations of expression levels called *expression maps*.

Another challenge addressed is the integration of morphological and expression data collected for multiple wings. The individual structure models inferred for each wing using the unsupervised detection and segmentation method are integrated in a single structure model. The procedure required information about the orientation of each structure model which has been previously identified from *a priori* information about the morphology of the pouch. The resulting model represents the average shape of the individual models and is always presented in a canonical orientation (anterior to left, dorsal to top). Moreover, the coordinate systems derived from the individual structure models enabled the systematic collection of expression datasets from many wings. The obtained individual expression maps are averaged to produce a representative expression map. Mean expression maps are then combined with the associated mean structure model in order to produce a single and robust quantitative description of the *Drosophila* wing pouch (**Fig. 4.42**). This description can then be extended with the output of an unsupervised cell nuclei detection and segmentation that we have developed (**Video S7**).

A large amount of effort has been expended to make the above methods fully automated. We have implemented them as part of an extensible, user-friendly, and open source image processing toolkit called WingJ. After providing WingJ with the stacks of confocal images, the quantification of the morphology and expression information of individual *Drosophila* wings (**Video S5**) and *Drosophila* embryos (**Video S6**) are achieved with only a few clicks. WingJ also allows the user to interact with the inferred model, for example if manual fine-tuning is required. Moreover, the integration of the structure and expression datasets collected in multiple wings can be performed directly in WingJ. We have applied our method to quantify hundreds of *Drosophila* wings imaged at different time points during development. The data

collected have then be used to generate quantitative descriptions of the developing wing. Furthermore, the generation of these quantitative descriptions provides a powerful tool to assess the effect of mutations on the morphology and domains of expression of genes. As an example, we have applied our method to generate quantitative descriptions of wild type and *pent* deficient wings, and have shown how the $pent^{2\text{-}5}$ mutation inhibits the growth of the wings. Finally, we have used the quantitative descriptions generated to reverse engineer a six-gene regulatory network that participates to the development of the *Drosophila* wing.

5 Discussion and outlook

In this chapter, we summarize the main contributions of this thesis and discuss implications for future research. In particular, we discuss the generation of multicellular *in silico* benchmarks for reverse engineering algorithms capable of predicting the formation of spatial gene expression patterns. We also suggest possible ways of improvement for community structure detection in complex networks. Finally, we provide hints for the extension of the framework that we developed for unsupervised and systematic segmentation of biological organisms. We expect that our approach will be extensively used in the future for providing the tremendous amount of data required to enable the reverse engineering of multiscale models.

5.1 Main accomplishments

The objective of this thesis was to develop and implement a comprehensive method for reverse engineering gene networks. Over the last decade, numerous methods have been developed for inference of regulatory networks from gene expression data, however relatively little effort has been put into evaluating the performance of those methods on adequate benchmarks.

The first contribution of this thesis is the generation of biologically plausible *in silico* benchmarks for performance profiling of network inference methods, and its implementation as a user-friendly and opens source software called *GeneNetWeaver (GNW)*[a]. In addition to the generation of detailed dynamical models of gene regulatory networks to be used as benchmarks, GNW provides a network motif analysis that reveals systematic prediction errors, thereby indicating potential ways of improving inference methods. The accuracy of network inference methods is evaluated using standard metrics such as precision-recall and receiver operating characteristic (ROC) curves. We show how GNW can be used to assess the performance and identify the strengths and weaknesses of six inference methods. Furthermore, we used GNW to provide the international DREAM (Dialogue for Reverse Engineering Assessments and Methods) competition with three network inference challenges (DREAM3, DREAM4, and DREAM5). Today, the accuracy of more than 25,000 gene network reconstructions have been evaluated by researchers and computer scientists using GNW.

The primary goal of reverse engineering algorithms is to reconstruct the complex network of interactions between the genes and their RNA and protein products from expression data. In addition, a fraction of these algorithms enable the inference of an *in silico* network model that can be used to predict the response of the network to new perturbations. A second contribution of this thesis is the rational decomposition of the predicted networks into functional modules, hence providing additional insight into the functions and mechanisms performed by the network. The detection of these modules, also referred to as *community structure detection*, has been largely addressed. However, it has been only recently that suitable benchmarks have been proposed to reliable evaluate the performance of community structure detection methods. In particular, it has been shown that the performance of modularity optimization methods, which is by far the most common approach used, is affected by a resolution limit that makes them fail to identify small communities in small networks. Here, we have described a novel GA-based community structure detection algorithm and have evaluated its performance on real and artificial networks. To overcome the resolution limit that has been shown to affect modularity-based methods, we have developed a voting method that enable the integration of multiple community structure predictions into a single and more reliable partition of the network into modules. In addition to provide a way to potentially overcome the weakness of many algorithms that are affected by this resolution limit, the GA-based algorithm followed by the voting method is best performer along with another method in a comparative analysis that profiled the performance of twelve state-of-the-art community structure detection algorithms.

[a] tschaffter.ch/projects/gnw

Another contribution of **Chapter 3** is the development and implementation of an extensible and modular software for community structure detected called *Jmod*[b]. Jmod implements state-of-the-art community structure detection methods including Newman's spectral algorithm, the GA-based modularity optimization method that we developed, and a brute force approach. It also includes two refinement techniques called *moving vertex method (MVM)* and *global moving vertex method (gMVM)*. Moreover, the extensible framework implemented in Jmod supports the development and integration of novel community structure detection methods. Jmod also provides tools to gain insight into the behavior of these methods, thereby indicating potential ways of improving them.

So far, only a small number of developmental systems have been reverse engineered[171–173]. One of them is the gap gene network[41] which is involved in segment determination during the early development of the *Drosophila* embryo[39,148]. Compared to single-cell systems, the reconstruction of developmental systems requires the development of multiscale models that account for processes at the molecular, cellular, and tissue level. Another limitation is that the inference of such models requires tremendous amount of spatial and temporal gene expression data, which are usually available in very limited quantities due to the inherent difficulty in measuring gene expression in an entire organism.

The third main contribution of this thesis is the development of an image processing application called *WingJ*[c] for fully automated and systematic quantification of the developing *Drosophila* wing, which is a classical model for studying the genetic control of tissue size, shape and patterning. All that is necessary for our method is a stack of confocal fluorescence images (3D image) of the biological system to quantify. First, a parametric model of the morphology or structure of the *Drosophila* wing is inferred from a fluorescent marker. The segmentation method is based on the design of multiple image processing detection modules, each focusing on the extraction of a specific feature of the wing structure including its orientation. We later extended this approach to the detection of the *Drosophila* embryo. We then use the inferred structure model as a convenient coordinate system for measuring gene and protein expression levels. An important feature of the obtained expression maps is that they can be used to compare domains of expression in differentiated systems, for example to visualize the difference in patterns of gene activity between wild type and mutant wings or in wings imaged at different time points during development. Moreover, a robust, multiscale quantitative description of the developing wing is obtained by combining morphological and gene expression information from multiple wings, completed by the output of an automatic cell nuclei detection method that we have developed. We have used the above method to automatically generate robust quantitative descriptions of wild-type and *pent* deficient *Drosophila* wings imaged at 80, 90, 100, and 110 hours after egg laying. Finally, we have shown that these quantitative descriptions can be used to unravel the regulatory interactions of a six-gene network involved in the development of the *Drosophila* wing.

[b] tschaffter.ch/projects/jmod
[c] tschaffter.ch/projects/wingj

5.2 Future directions

5.2.1 Generating *in silico* developmental benchmarks

Using GeneNetWeaver, one can intuitively generate biologically-plausible *in silico* (or *virtual*) gene regulatory network to be used as benchmark to evaluate the performance of its inference method. Network topologies are generated by extracting modules from known *in vivo* gene regulatory network structures such as those of *E. coli* and *S. cerevisiae*. These structures are then endowed with detailed dynamical models of gene regulation including both transcription and translation processes using a thermodynamic approach accounting for both independent and synergistic interactions. Expression data can be generated either deterministically or stochastically to model molecular noise in the dynamics of the networks, and experimental noise can be added using a model of noise observed in microarrays. Different types of *in vivo* experimental procedures, such as wild type, knockout (null-mutant), knockdown (heterozygous), and multifactorial perturbations, can be reproduced by the software.

Moreover, we have investigated the generation of multicellular benchmarks for assessing the performance of developmental network inference methods. The model is based on the use of *morphogens*, which are transcription factors that diffuse outside of the cells[44,172]. There are two types of morphogens: long-range and short-range. Long-range morphogens are generally produce by a group of cell and diffuse through the whole tissue[146,174]. Short-range morphogens are produce at different places and diffuse only in a small region of the tissue[175]. Their activity gradients then provide cells with positional information required for cell fate determination. Depending on the concentration of a morphogen available, cells absorb different amount of morphogen molecules which then lead to differential gene activation and cell differentiation.

Most of the small number of inference methods for the reconstruction of developmental gene networks are for now restrained to one dimensional spatial models. The model that we have started to investigate is given by[145]

$$\frac{\delta c}{\delta t} = D\frac{\delta^2 c}{\delta x^2} - \lambda c + s_0 p(x) + k_r C_i - k_a c N \tag{5.1}$$

whose terms represent the concentration of the morphogen $c(x,t)$, free diffusion (Fick's second law of diffusion), the degradation of morphogen, its production from a source, and the rejection and absorption of the morphogen by the cells. Moreover, a common assumption made by developmental network inference methods is that morphogens propagate only through free diffusion[40,148]. This is most certainly correct when considering the early development of the *Drosophila* as cell membranes have not yet been fully formed. However, different passive and active mechanisms have been proposed to describe the formation of morphogen activity gradients. The active transport mechanisms implemented in (5.1) is called *transcytosis* and consists in morphogen molecules that move in the tissue through *endocytosis* (absorption by

the cells) and *exocytosis* (rejection by the cells)[176,177]. Other models suggest that glypicans (small *hairs* on the cells) also play a role in the active transport of morphogen molecules in addition to protect them from degradation[178,179].

Models developed for *in silico* benchmark generation are in principle more detailed than those that are reverse engineered. For example, most of the inference methods that attempt to capture the dynamics of regulatory networks use linear models when many important biological questions absolutely require the consideration of non-linear systems. The inference of non-linear models is actually a considerable technical challenges[180–182]. The principal reason is that the inference problem is typically underdetermined due to the lack of sufficient input datasets. As a consequence, a stochastic inference method may predict several network models that describe the given observation instead of unravelling systematically the true regulatory network. Therefore, novel methods must imperatively be developed to enable the collection of large and heterogeneous datasets from the system to reconstruct.

5.2.2 Community structure detection in complex networks

In this thesis, we have developed several community structure detection methods that enable the inference of the partition of the network into modules. These methods are an improved version of Newman's spectral algorithm, a genetic algorithm-based method, and a brute force approach. These three methods, as well as most of the methods developed by the community, are based on the optimization of a metric called *modularity*[26,27].

A limitation of these methods is that they usually do not take into account information provided by the direction and weight of the interactions. Nevertheless, relationship and interaction networks have often edges with precise direction which must be taken into account to understand the system as a whole[76]. The original definition of the modularity Q proposed by Girvan & Newman is limited to undirected and unweighted graphs[26]. In this thesis, we have used a second definition proposed by Newman which models the weights of the interactions[27], but not their directions. The modularity Q has been previously described as

Q =(fraction of edges falling within modules)
- (expected fraction of such edges in randomized graphs)

Here we consider a directed graph and two vertices A and B. A has a high out-degree (number of edges starting from A) and B has a high in-degree (number of edges arriving to B). When observing the graph, the probability of finding an edge that runs from B to A should be lower than the reverse situation. Therefore, the contribution of the edge $B \rightarrow A$ to the modularity Q should be larger than that of $A \rightarrow B$ since "modularity should be high for statistically surprising configurations"[27]. An extension of Q has been proposed by Leicht & Newman to integrate the information included in directed networks[183]. However, the adaptation to directed networks

is not that easy and only a few techniques can use the above extension, including our GA-based detection method. The adaptation of spectral algorithms, such as the one developed by Newman, is much more complex as the adjacency matrix that characterizes a directed graph is now asymmetric.

Moreover, we suggest that the development of ensemble methods is one of the most promising way to significantly improve the reliability and robustness of community structure detection. Ensemble methods is an approach that is widely used in the field of machine learning[184,185]. However, it has been only recently that these methods have found other domains of application including the reverse engineering of gene networks[4]. A similar approach has been applied by Lancichinetti & Fortunato to the problem of community structure inference[102]. From multiple partitions of the network, usually predicted using different methods, they evaluate for each edge the fraction of time that is has been predicted to fall entirely inside a module (in opposition to having its two nodes predicted to belong to two different communities). A consensus matrix is built from these values and represented as a weighted consensus graph, which is usually full connected. An arbitrary threshold is then applied to remove the weakest edges. Finally, one of the existing community structure detection algorithms is applied on the thresholded consensus graph to obtain a new and more reliable partition of the original graph.

This method has two disadvantages. First, useful information is discarded when thresholding the consensus matrix. The second issue is that the method still relies on a single community structure detection algorithm to partition the consensus graph. Instead, we propose to use the community voting method introduced in **Chapter 3** to combine multiple partitions of the same network into a single partition which we have shown to be more reliable and robust than individual partitions. The principal difference with the previous method is that we consider nodes rather than edges as the building blocks of the communities. Furthermore, our method does not require any parameter values to be set.

5.2.3 Unsupervised detection and segmentation of biological organisms

In this thesis, we have developed a novel method for unsupervised and systematic segmentation of biological systems, which we have implemented as an extensible and user-friendly image processing software. First, we generate a parametric model that accurately describes the morphology or structure of the system to model (organ or body system). One of the novelty of the approach is that the quantification of the structure is decomposed into the detection of simpler morphological features. The model that describes the structure as a whole is then reconstructed from the segmentation of these features. A great contribution of this modular approach is that the *detection modules* developed to segment simple features can be reused for generating parametric models of the morphological structure of other organisms. Once a large collection of detection modules has been implemented, the detection of any structure could potentially become as simple as selecting the different modules to apply.

Using the above approach, we have enabled the generation of quantitative descriptions of the *Drosophila* wing pouch, which is a classical model for studying the genetic control of tissue size, shape and patterning. Early in development, the wing pouch is composed of a single-layered sheet of columnar cells (**Fig. 1.1**), hence enabling its description as a two-dimensional spatial system. It is only later at late third instar (about 110 hours after egg laying) that the sheet of cells folds to form the double-layered adult wing (**Video S4**). Thereafter, one of the natural evolution of our method will pass through the development of three-dimensional segmentation algorithms. Actually, only very few image processing algorithms that have been developed for 2D images have been extended to work with 3D images. Nevertheless, we expect that more and more image processing algorithms will be available for the segmentation of 3D images[186]. Furthermore, the segmentation of 4D images with temporal information also holds huge promise. This will be one day possible with the constant improvement brought to live imagining technologies[47,187].

Even unsupervised methods usually relies on suitable parameter settings. For example, the detection and segmentation of the *Drosophila* wing pouch required the definition of many parameters. This enables the fine control of the detection algorithms, which can then be adapted to different situations. However, this becomes an issue if many parameters must be constantly updated to ensure the functioning correctly of the algorithm. Another important feature expected from such algorithms is their ability to perform systematic detection. When several experiments perform manual labelling, differences can usually be observed in the data they have collected. Unsupervised methods provide a way to overcome this limitation. Here, we propose a strategy to improve both the unsupervised and systematic behavior of the detection and segmentation method. Using the default parameter values of the software, the user first performs a few detections and correct the output of the detection when required. This feature is already implemented in WingJ and allows to intuitively fine-tune any identified parametric structure models. The idea is then to make the software *learn* the parameter values that would enable the unsupervised algorithm to return parametric structure models similar to the one defined by the user. This could be achieved using an optimization algorithm such as an evolutionary algorithm to minimize a fitness function defined as the difference or distance between the evolved structure models and the models validated by the user.

5.3 Conclusion

In this thesis we present a comprehensive and efficient framework for reverse engineering gene networks. The reconstruction starts with the collection of stacks of fluorescence images (3D images) where the expression of genes is labelled using fluorescent markers. Unsupervised detection and segmentation methods are then applied to generate a multiscale quantitative description of the system. This description provides information about the morphology of the system, the expression of a few genes of interest, and the location of cell nuclei. The quantitative description is then given as input to a reverse engineering algorithm to produce an *in silico* models that explain the observations. In addition, we have developed a method

for *in silico* benchmark generation and performance profiling of network inference methods. The approach consists in generating biologically plausible models of gene networks before simulating them to produce datasets similar to the quantitative description generated for the *in vivo* system to reconstruct. The performance of the inference methods are then evaluated using tools we provide before selecting the best method(s) to apply for the effective reconstruction of the system. Furthermore, an additional layer of reconstruct is applied to rationally decomposed the gene network into modules, thus providing insight into the information processing it performs.

One of the main achievements of this thesis is the development of unsupervised methods and their implementation as modular and user-friendly software applications. Science fiction novels and movies certainly provided a good deal of inspiration as the result of any detection or analysis is here systematically obtained in a few *clicks*. The proposed framework has been implemented to reverse engineer gene networks, however it has been designed keeping in mind that it may be used one day to enable the reconstruction of more complex systems. Furthermore, additional work would be required to propose an integrated solution that would perform at once quantification of biological systems and their reverse engineering.

A Supplementary materials

Project websites

For each project presented in this thesis, we have published a website that provides the software application, its source code, user manual, videos, and additional supporting data.[a] The Java applications GNW, Jmod, and WingJ can be launched with a single click directly from their websites.

- GeneNetWeaver (tschaffter.ch/projects/gnw)
- Jmod (tschaffter.ch/projects/jmod)
- WingJ (tschaffter.ch/projects/wingj)
- libSDE (tschaffter.ch/projects/libsde)
- sQuid (tschaffter.ch/projects/squid)

Videos

Video S1: *GeneNetWeaver: In silico benchmark generation and performance profiling of network inference methods.* This video reviews the features of GNW including the extraction of subnetworks from known transcriptional networks, the generation and simulation of the dynamics of the *in silico* network, and the generation of reports for performance profiling of inference methods.

Video S2: *Jmod: An extensible toolkit for community detection in networks.* This video illustrates community structure detection in social, biological, and artificial networks using different methods available in Jmod. The methods illustrated are Newman's spectral algorithm,

[a] Also available at tschaffter.ch/phd/documents.

our GA-based modularity optimization method, the brute force approach, and the refinement technique MVM and gMVM.

Video S3: *Development of the Drosophila wing.* This video provides a brief introduction to the development of the *Drosophila* wing using 3D rendering of confocal fluorescence images and a 3D animation which shows how the single-cell layered wing imaginal disc everts to give rise to the double-layered adult wing.

Video S4: *WingJ: Towards unsupervised and systematic segmentation of biological systems.* This video features the different algorithms we have developed and implemented in WingJ for unsupervised segmentation of the *Drosophila* wing pouch and *Drosophila* embryo. This includes automatic segmentation of the morphology of the *Drosophila* wing and embryo, gene and protein expression quantification, and unsupervised cell nuclei detection from stacks of confocal fluorescence images. *Additional credits: Ricard Delgado-Gonzalo.*

Video S5: *Unsupervised detection of the Drosophila wing pouch.* This video provides an example of the generation of a quantitative description of the *Drosophila* wing pouch. First, morphological and gene expression data are collected from multiple wings using the unsupervised method we developed. The datasets are then combined to generate a robust and reliable quantitative description of the pouch.

Video S6: *Unsupervised detection of the Drosophila embryo.* In the same way as the previous video, the present video shows how to use WingJ to generate quantitative description of the *Drosophila* embryo.

Video S7: *Automatic cell nuclei detection in structure models.* This video shows the output of our fully automated cell nuclei detection method when applied to the *Drosophila* wing pouch system. The method is based on a 3D watershed transform and takes as input 1) a stack of confocal images where nuclei have been labelled with a fluorescent dye and 2) the structure model of the system previously inferred using WingJ to define the space in which cell nuclei must be segmented. Thus, the approach can be used either to collect cell nuclei information in a complete organism or only in a part of interest or organ.

Video S8: *sQuid: Observation and interaction in experimental environments.* This video shows how we track the genetic identity of fruit flies walking in a transparent chamber. sQuid is used to control two FireWire cameras, two LEDs to visualize the expression of different genes, and several valves that regulate odor flows in the arena to interact with the flies.

B Supplementary notes for Chapter 2

B.1 GeneNetWeaver

B.1.1 Topology

We generated network structure by extracting modules from the biological interactions networks of *E. coli*[6] and *S. cerevisiae*[5] (the *source networks*). The benchmark suites *A*, *B*, and *C* contain 100-, 200-, and 500-gene networks. For each network, we set the minimum number of regulators (nodes with at least one outgoing link *in the source network*) to half of the size of the extracted network. The parameter *seed* has been set to *random vertex* and *neighbor selection* has been set to *random among top 50%*. Networks with more than one connected component were discarded.

B.1.2 Dynamical model

Network topologies are endowed with detailed dynamical models of gene regulation. Both transcription and translation are modeled using a standard thermodynamic approach[53] allowing for both independent ("additive") and synergistic ("multiplicative") regulatory interactions. A detailed description of the dynamical model used is given by Marbach *et al.*[62]. First, auto-regulatory interactions were removed from the extracted networks before setting the dynamical model. The most important parameters are the *mRNA* and *protein half-lives* in minutes sampled from a Gaussian distribution $\mathcal{N}(27.5, 56.25)$ bounded in the interval $[5, 50]$, the *dissociation constants* sampled from a uniform distribution $[0.01, 1]$, and the *Hill coefficients* sampled from a Gaussian distribution $\mathcal{N}(2, 4)$ bounded in the interval $[1, 10]$.

B.1.3 Synthetic expression datasets

The next step in generating *in silico* benchmark networks consists in simulating the generated *in silico* regulatory networks to produce synthetic gene expression datasets. Initially, $\boldsymbol{x}$ and $\boldsymbol{y}$

are set to null vectors in (2.1) and (2.2). They two equations are then simulated until a steady state is achieved, at which point $\boldsymbol{x}$ and $\boldsymbol{y}$ are measured and saved as initial conditions for future perturbation experiments. Systematic knockout and knockdown experiments were simulated to generate steady-state expression data. Also, 100 multifactorial perturbation experiments were simulated to generate steady-state expression data for each network from the benchmark suite C. The parameters were set to the same values used to generate the DREAM4 *In Silico* Challenge we provided. Those settings also correspond to the default parameter values provided by GNW. More specifically, we modeled molecular noise with the *coefficient of (molecular) noise term* set to 0.05[51], in addition to a model of experimental noise observed in microarrays[54].

B.1.4 Gold standards and network prediction format

Performance profiling of network inference methods using GNW requires *gold standards* and *network predictions* files to be provided. The gold standard files can be imported to GNW using either the format TSV, GML, DOT, or SBML[a]. The gold standard files in TSV format must be formatted as follows

```
G0   G1   1
G0   G2   1
...
G1   G0   0
...
```

Each line defines an interaction oriented from the first gene to the second gene. The third element is 1 if the interaction is present in the gold standard and 0 otherwise. Instead of listing the absent (0) interactions, they can also simply be omitted. The format for the predictions is the same as used for the DREAM challenges

```
G0   G1   0.98
G0   G2   0.8
...
G1   G0   0
...
```

As in the gold standard file, each line defines an interaction oriented from the first to the second gene. For each interaction, a *confidence level* between 0 and 1 is given that indicates the degree of belief that the interaction is included in the gold standard. The predictions must be listed in *descending* order relative to their confidence level (the first prediction in

[a] In the current version, only SBML files that have been generated by GNW can be opened.

the list being the most confident). The confidence levels are only used to verify that the list of predictions is correctly ordered, they do not affect the PR and ROC curves and the motif analysis in any other way.

B.1.5 Evaluation of network inference methods

From a set of predictions from one or several inference methods, GNW automatically generates a comprehensive report including the result of a network motif analysis, where the performance of inference methods is profiled on local connectivity patterns (network motifs). The network motif analysis often reveals systematic prediction errors, thereby indicating potential ways of network reconstruction improvements[62]. Furthermore, precision-recall (PR) and receiver operating characteristic (ROC) curves are evaluated for each network prediction[66]. The relation between ROC and PR curves is discussed by Davis[67]. The intuitive interface of GNW allows to easily evaluate several inference methods at a time to facilitate the comparison of their relative performance. Evaluation results are always saved in a text file (XML format). In addition, GNW can generate PDF reports with plots from these data (an internet connection is required). Without internet connection, the evaluation can still be run but no PDF report will be created.

B.2 Network inference methods

B.2.1 Z-score

Z-score is one of the simplest inference methods[66], yet it has relatively high accuracy in predicting directed network structures from knockout steady states (see Section 3.2 of the paper). For each gene of a network, Z-score computes the mean μ and standard deviation σ of the gene expression level from several experiments. Then for each single-gene knockout perturbation, a regulatory interaction is identified if the measured expression level of a given gene is below $\mu - \sigma$ (enhancing regulation) or above $\mu + \sigma$ (inhibitory regulation). The Matlab implementation of Z-score used is the one provided by Pinna *et al.*[70]. Z-score does not require any parameters to be set.

B.2.2 Pinnal *et al.*

The algorithm developed by Pinna *et al.* allows to choose between four possible different confidence matrices W to obtain the initial predictions[70]. Here Z-score is applied on the raw gene expression data to generate the initial predictions (W^{ZR}). Then the method performs a refinement stage, which aims to suppress the errors made by Z-score on cascade motifs from knockout steady states. This improvement is achieved by reducing the confidence initially predicted to unnecessary feed-forward edges[70]. The parameters used are the default ones. Especially, the threshold parameter t is set to 2 ($t = 0$ corresponds to not applying

the refinement stage, i.e. Z-score alone), which is also the value[70] used to participate to the DREAM4 *In Silico* Challenge Size 100. Pinna *et al.* was best-performer in that challenge.

B.2.3 Yip *et al.*

The original Java tool developed by Yip *et al.*[71] implements different techniques to infer gene regulations from both steady-state and time-series data. From steady-state expression data, a noise model is learnt to distinguish real signals from random fluctuations (*Batch 1*). Ordinary differential equations (ODE) are then used to model the change of expression levels of a gene along the time series due to the regulation of other genes[71]. Yet, Yip *et al.* applied their noise model alone to participate to the DREAM3 *In Silico* Challenge we provided, and their method was best-performer in all sub-challenge of size 10, 50, and 100 genes. Here the noise model was applied alone to predict directed networks from knockout expression data. This part of the method developed by Yip *et al.* does not require any parameters to be set.

B.2.4 CLR

The *context likelihood of relatedness (CLR)* algorithm developed by Faith *et al.*[58] is an unsupervised network inference method using mutual information as a metric of similarity between the expression profiles of two genes. The method does not require systematic knockout gene expression data, which are not always available in practice, to infer undirected networks. We applied CLR 1.2.2 using the provided binary for Linux. All mutual information values were computed using *5 bins* and *third order B-splines*[58].

B.2.5 ARACNE2

Similar to CLR, the inference method developed by Margolin *et al.*[57] also uses mutual information as a metric of similarity between the expression profiles of two genes. The method allows the reconstruction of undirected networks from steady-state expression data, and does not require systematic knockout or knockdown experiments. The Java implementation of ARACNE2 was used with the provided default settings, that is, the *algorithm* set to *fixed_bandwidth*, the *p-value for MI threshold* to 1, the *DPI tolerance* to 1, the gene filter configured with *mean* and *cv* to 0, and the *MI threshold* to 0.

B.2.6 GENIE3

Huynh *et al.* decomposes the prediction of a regulatory network between p genes into p different regression problems[188]. GENIE3 has the potential ability to predict directed networks, while methods based on mutual information or correlation can only predict undirected networks unless additional information is used. The Matlab implementation of GENIE3 was

used with the *Random Forests* procedure, the parameter K set to the square root of the number of input genes, and the number of trees grown in an ensemble set to 1000[188].

C Supplementary notes for Chapter 3

C.1 Eigendecomposition

In linear algebra, the eigendecomposition of a matrix A consists in representing a matrix in terms of its eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ and associated eigenvectors $v_1, v_2, ..., v_n$. Eigenvalues are usually ordered so that $|\lambda_i| > |\lambda_{i+1}|$ for $i = 1, 2, ..., n-1$. The eigendecomposition requires that the matrix to decompose is squared and diagonalizable, that is if there exists an invertible matrix V such that $V^{-1}AV$ is a diagonal matrix. Therefore, the objective of the eigendecomposition is to find the matrices D and V satisfying

$$A = V^{-1}DV \tag{C.1}$$

where D is a diagonal matrix containing the eigenvalues of A and where the columns of V are the associated, linearly independent eigenvectors.

C.1.1 Power method

The *power method* or *power iteration* is a simple scheme to approximate the largest eigenvalue (in absolute value) and the associated eigenvector of a real matrix but not necessarily symmetric[189,190]. This eigenvalue is usually called the dominant or leading (most positive) eigenvalue. In many applications, this quantity must be positive for physical reasons. For example, Google uses the power method to rank the most important web pages[191,192].

In Newman's spectral algorithm, the power method is used for finding the dominant positive eigenvalue λ_1 and associated eigenvector v_1 of the real symmetric dense modularity matrix $\mathbf{B}$ (or generalized modularity matrix $\mathbf{B}^{(\mathbf{g})}$). **Algorithm C.1** gives the pseudo code of the power method as implemented in Jmod.

The parameter $maxIters$ is the maximum number of iterations and ϵ is the precision for

Algorithm C.1: power method

Data: modularity matrix **B**, $maxIters$, precision ϵ
Result: dominant eigenpair (λ_1, v_1)
$q^{(0)} \leftarrow$ random vector $\in \mathbb{R}^n$
$k \leftarrow 1$
repeat
  $z^{(k)} \leftarrow \mathbf{B}q(k-1)$
  $q^{(k)} \leftarrow z^{(k)}/||z^{(k)}||$
  $\lambda^{(k)} \leftarrow \left[q^{(k)}\right]^T \mathbf{B}q^{(k)}$
  `/* Update convergence criteria */`
  $\phi \leftarrow 0$
  for $i \leftarrow 1$ *to* n **do**
    $\Delta \leftarrow q_i^{(k)} - q_i^{(k-1)}$
    if $\Delta > \phi$ **then**
      $\phi \leftarrow \Delta$
    end
  end
  $k \leftarrow k+1$
until $\phi > \epsilon$ *and* $k < maxIters$;
$\lambda_1 \leftarrow \lambda^{(k)}$
$v_1 \leftarrow q^{(k)}$

convergence. The maximum number of iterations and the parameter ϵ must be carefully set. Ill adapted values can result in premature convergences or spending unnecessary time to satisfy unreasonable constraints. At that point, it is interesting to note that only the sign of the elements in the dominant positive eigenvector are used to define the split vector **s**. This means that even if the required precision ϵ is not met yet, the current vector $q^{(k)}$ can already be used to define the split vector **s** as long as the sign of its elements does not change.

In order to set suitable values to the maximum number of iterations and the required precision ϵ, we first set them respectively to the demanding values 10^6 and 10^{-10}. Newman's spectral algorithm is run using the power method to perform the eigendecomposition of the modularity matrix **B** (or the generalized modularity matrix $\mathbf{B}^{(g)}$ for further split of a community in two) of twenty 1000-node LFR graphs with including small communities taken and for three different values of the mixing parameter μ (see **Section 3.4.1**). As a reminder, smaller values of μ lead to more modular graphs. For each split of the graphs, the iteration index k and the corresponding current precision ϕ are saved the *last time* that at least one element of the vector $q^{(k)}$ sees its sign flipped, that is when one node is moved from one subcommunity to the other (**Section 3.2.1**). **Figure C.1** shows the number of iterations required to find the "correct" sign of all the elements in $q^{(k)}$ (as returned once the demanding constraints previously set are satisfied) against the size of the community split.

The largest number of iterations required is $4.985 \cdot 10^4$ (the maximum was set to 10^6) over

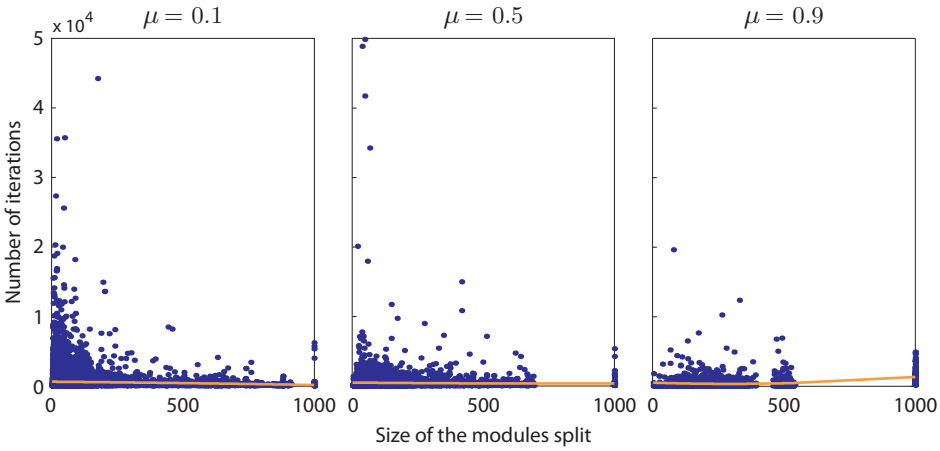


Figure C.1: Evaluation of the power method to determine suitable stopping criteria. Newman's spectral algorithm uses the power method[189,190] to compute the dominant positive eigenvector v_1 which defines entirely the split vector **s** that describes the split of a community in two. We select three 1000-node LFR graphs for different values of the mixing parameter μ which controls the effective modularity of the LFR graphs (small μ leads to more modular graphs). Each blue dot corresponds to the number of iterations that the power method reached the last time an element of the eigenvector v_1 has seen its sign flipped. The two plots on the left may look like the smaller the community to split requires the more iterations of the power method to converge. This is actually not the case as shown by the second order fits (in orange). The maximum number of iterations reached over 18'463 community splits is $4.985 \cdot 10^4$. For the experiments reported in **Chapter 3**, we decide to set the maximum number of iterations to 50′000.

18'463 community splits. For $\mu = 0.1$ and $\mu = 0.5$, one may conclude that higher number of iterations are required to split smaller communities. This is actually not the case and originates from the fact that there are way many splits of small communities than large communities. This is confirmed by a second order fit (in orange). In addition, we observe that for $\mu = 0.9$, i.e. graphs with low modularity, most if not all graphs are first split in two communities that include each about 500 nodes, before being further divided in smaller communities. Another evidence of the effect of μ is that 10'925, 5'035 and 2'503 splits were performed using Newman's spectral algorithm for μ set to 0.1, 0.5 and 0.9, respectively.

The obtained values of ϕ are reported with a median of 0.001 and a 95% confidence interval (CI) of $1.569 \cdot 10^{-5}$ to 1.999. The later boundary, which seems very large, originates from the fact that the elements of $q^{(k)}$ range from -1 to 1 and so flipping the sign of an element with absolute value 1 leads to $\phi = 2$.

From the above observations, we choose to set the maximum number of iterations to $5 \cdot 10^4$ and the required precision ϵ to 10^{-5} to ensure the converge of almost all runs and at the same time to prevent against spending unnecessary time in the power method, which already takes

most of the computation time required to run Newman's spectral algorithm.

Note that the dominant eigenvalue computed using the power method is not necessarily positive, which is a requirement to obtain a positive contribution to the modularity Q. If the dominant eigenvalue is positive, the eigenvector v_1 can be used directly to obtain the split vector **s**. If the dominant eigenvalue is negative, a second chance is given to find the dominant positive eigenvalue using a simple trick and the output of the power method. First, the shifted matrix $(\mathbf{B} - \lambda_1 I)$ is computed (I is the identity matrix), which has eigenvalues $\lambda_i - \lambda_1$ all positives (here λ_1 is negative) but conserves the eigenvectors of **B**. The power method is then run a second time on $(\mathbf{B} - \lambda_1 I)$ to hopefully obtain the eigenvector associated to the dominant positive eigenvalue. If it does not exist or is not found, the community being currently partitioned is considered as indivisible. More precisely, a property of **B** is that the elements of each of its rows and columns sum to zero and so **B** has always the eigenvector [111...] with eigenvalue zero.

This power method converges to the leading eigenvector for a running time of $O[(m+n)n]$ overall (m is the number of edges in the network) or $O\left(n^2\right)$ on a sparse graph with $m \propto n$[27]. It is important to note that this accounts for the split in two of a single community when the communities of a graph are recursively split in two. As described previously, this running time can double in cases where the dominant eigenvalue computed is negative. Finally, it has been shown that the power method converges at a rate that is equal to the ratio of the two largest eigenvalues $\left|\frac{\lambda_2}{\lambda_1}\right|$ and thus converges slowly if there is an eigenvalue close in magnitude to the dominant eigenvalue[193,194].

C.1.2 Lanczos algorithm

Newman's spectral algorithm makes originally use of the power method to find the dominant positive eigenpair $(\mathbf{B} - \lambda_1 I)$ associated to a modular decomposition[27]. Here we were interested in comparing the result of the power method with another approach to approximate the eigenvalues and eigenvectors of a matrix. The method we retained is the *iterative Lanczos algorithm*[195,196] implemented in COLT, which is a set of open-source libraries for high performance scientific and technical computing in Java. An implementation of the Lanczos algorithm is also contained in Matlab and GNU Octave.

The Lanczos algorithm is an adaptation of the power method described in **Section C.1.1** and is particularly effective for decomposing very large sparse matrices[196]. Note that the Lanczos method as it is implemented in the COLT library approximates all eigenpairs together. Yet another approach would consist in combining both the power method and the Lanczos algorithm to only find the dominant eigenpair[197]. The complexity of the Lanczos method seems largely dependent on the applications and is difficult to estimate, although hints are given by Arora *et al.*[198]. Nevertheless, we expect the Lanczos algorithm to have a complexity $O\left(n^2\right)$ similar to the complexity of the power method.

C.1.3 Evaluation of the power method and Lanczos algorithm

We evaluated the running time of two eigendecomposition algorithms that can be used in Newman's spectral algorithm (**Section 3.2.1**) to identify the community structure of 1000-node LFR graphs (**Section 3.4.1**). The algorithms considered are the power method and the Lanczos algorithm provided by the COLT library[a] which can be used to compute the dominant eigenvalue λ_1 of the modularity matrix **B** and its associated eigenvector v_1.

The benchmark is composed of 1000-node LFR graphs made of small communities generated for different values of the mixing parameter μ which defines the modularity of the benchmark graphs. Low value of μ generate highly modular graphs. For each value of μ, we only perform the first split of one hundred graphs in two communities, that is, we compute only the first eigendecomposition of the n-by-n modularity matrix **B** for each graph.

Figure C.2 reports the computation time of the algorithms in milliseconds.[b] The average computation time is shown as well as the 95% confidence interval computed for the mean. The power method requires about half the time that the Lanczos algorithm needs to converge. Both methods take $O\left(n^2\right)$ time, but the power method is here faster because it only computes the leading eigenpair $\{\lambda_1, v_1\}$. The Lanczos algorithm takes more time but returns all the eigenpairs of **B**, which are not used in Newman's algorithm.

The power method converges at a rate that is proportional to the ratio of the two largest eigenvalues $\left|\frac{\lambda_2}{\lambda_1}\right|$ and thus converges slowly if there is an eigenvalue close in magnitude to the dominant eigenvalue[193,194]. In modular networks, the first eigenvalue of **B** is usually quite different from the second eigenvalue[27]. However, this difference is expected to decrease when the modularity of the graphs decreases (i.e. when μ increases). **Figure C.2** shows that the computation time required by the power method increases for $\mu \geq 0.8$, which support the fact that more time are required to separate the first and second eigenvectors of **B** in non-modular networks.

C.1.4 GA parameter values

Here we summarize the methods and parameter values of the genetic algorithm-based module detection method evaluated in **Section 3.4.3**. The performance of this algorithm and the other module detection methods and refinement methods introduced in **Sections 3.2** and **3.3** is evaluated in **Section 3.4.4** and following.

- **Genetic encoding:** Binary
- **Population size:** 100
- **Initialization:** Random individuals

[a] acs.lbl.gov/software/colt

[b] Intel® Xeon® CPU X5472 @ 3.00GHz

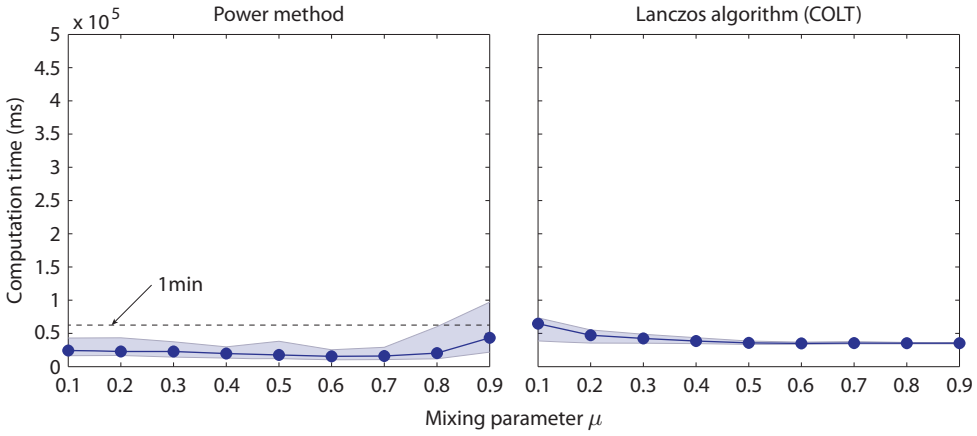


Figure C.2: Computation time of the power method and Lanczos algorithm. These two methods are applied to compute the leading eigenvector associated to the most positive eigenvalue of the modularity matrix $\mathbf{B}$ of 1000-node LFR graphs generated for different values of the mixing parameter μ. Low values of μ are used to generate highly modular graphs. Here we report the mean and the 95% confidence interval (CI) of the running time in milliseconds required by the power method and the Lanczos algorithm provided by the COLT library to compute the eigenvector v_1. Each point is computed from the application of the methods to 20 graphs. The results show that the power method requires in general less time than the Lanczos algorithm to converge on modular graphs.

- **Selection:** Tournament selection of size 2 (elitism set to 1)
- **Crossover:** Uniform crossover with rate set to 1
- **Mutation:** Bit-flip mutation with rate set to 1 bit mutated per genome
- **Stopping criterion:** GA stops when the population diversity (average Hamming distance) is less than 1 bit, i.e. when the difference between any pair of individuals is on average less than one node.
- **Maximum number of generations:** 3000
- **Others:** brute force method applied for splitting small communities

C.2 Pseudocode of MVM

The MVM takes as input a split vector $\mathbf{s} \in \mathbb{R}^c$ that defines the split of one community in two subcommunities as found by a module detection method. c is the size of the current community to split. The modularity Q_c and the modularity matrix $\mathbf{B}^{(c)}$ associated to the current community are also required. If this is the first division of the graph, Q_c is Q and $\mathbf{B}^{(c)}$ is the modularity matrix $\mathbf{B}$. Otherwise Q_c corresponds to ΔQ and $\mathbf{B}^{(c)}$ is a generalized modularity matrix $\mathbf{B}^{(g)}$ defined by Newman[27].

The idea behind the MVM is that one vertex is moved at a time in the split vector **s** to refine, which is the element i that contributes the most to increase Q_c. The algorithm then restarts until there are no more vertex moves improving the modularity. The details of the gMVM is given by **Algorithm C.2**.

Algorithm C.2: Moving vertex method (MVM)

Data: community size c, split vector $\mathbf{s} \in \mathbb{R}^c$, community modularity Q_c, community modularity matrix $\mathbf{B}^{(c)}$, total number of edge m, threshold θ
Result: refined split vector **s**, updated community modularity Q_c

```
/* Initialization                                                    */
currentQ_c ← Q_c
tmpQ_c ← 0
/* MVM                                                              */
repeat
    /* Index of the vertex moved                                    */
    i ← −1
    /* Index of the vertex tested                                   */
    for j ← 1 to c do
        sum ← 0
        for k ← 1 to c do
            sum ← sum + B^(c)_kj s_k s_j
        end
        sum ← sum − B^(c)_jj
        tmpQ_c ← Q_c − sum/m
        if (tmpQ_c − currentQ_c) > θ then
            currentQ_c ← tmpQ_c
            i ← j
        end
    end
    if i > 0 then
        s_i ← −s_i /* flip -1→1 or 1→-1                             */
        Q_c ← currentQ_c
    end
until i > 0;
```

C.3 Pseudocode of gMVM

Algorithm C.3 gives the procedure of the refinement technique gMVM introduced in **Section 3.3.2**. This method can be applied as the final stage of any modularity optimization methods.

The gMVM takes individually each node of a graph and place them successively in each indivisible community identified, before computing the associated contribution to the modularity Q for each move. If the contribution is negative, the node is left in its former community.

Algorithm C.3: Global moving vertex method (gMVM)

Data: number of nodes n, number of edge m, number of communities c, global split vector $\mathbf{s} \in \mathbb{R}^n$, modularity Q, modularity matrix $\mathbf{B}$
Result: refined global split vector $\mathbf{s}$, updated modularity Q

```
/* Initialization                                                    */
for i ← 1 to n do
    s_i ← index of the community containing the vertex i
end
newQ ← Q
/* gMVM                                                             */
repeat
    changeIndex ← −1
    changeNewCommunity ← −1
    /* Loop over all vertices                                       */
    for i ← 1 to n do
        oldCommunity ← s_i
        rowSum_i ← 0
        for j ← 1 to n do
            if s_i == s_j and i != j then
                rowSum_i ← rowSum_i + B_ij
            end
        end
        /* Loop over all communities                                */
        for k ← 1 to c do
            if k != oldCommunity then
                newRowSum_i ← 0
                for j ← 1 to n do
                    if k == s_j then
                        newRowSum_i ← newRowSum_i + B_ij
                    end
                end
                tmpNewQ ← Q + (newRowSum_i − rowSum_i)/m
            end
            if tmpNewQ > newQ then
                newQ ← tmpNewQ
                changeIndex ← i
                changeNewCommunity ← k
            end
        end
    end
    if changeIndex > 0 then
        s_changeIndex ← changeNewCommunity
        Q ← newQ
    end
until changeIndex > 0;
```

Otherwise, the node is moved to the community associated to the largest contribution to the modularity Q. This operation is then repeated as long as there is a node move that increases the modularity.

C.4 Computation time of network module detection methods

C.4.1 Improved version of Newman's algorithm and GA-based method

We compare the computation time required by Newman's spectral algorithm and the GA-based method on modular and non-module LFR graphs (**Section 3.4.1**). Newman's algorithm is deterministic and uses the power method to compute the leading eigenvector of the modularity matrix **B**, which is then used to obtain the split vector **s**. The GA-method uses the parameters given in **Section C.1.4**. Both methods are run on the same desktop computer (Intel® Core® i7-3770K CPU @ 3.50GHz). The GA-base method uses 6 cores to speed up the detection process. The computation time of the method can largely be reduced using more processors. The implementation of Newman's algorithm is not parallel so it is possible to process multiple networks in paralllel in Jmod. However, this does not help if there is only one network whose community structure must be identified. **Figure C.3** shows the computation times of the two methods successively refined using MVM and gMVM.

Newman's algorithm is the fastest method as expected. MVM takes usually as much time as Newman's algorithm itself because Newman's method returns a broad approximation of **s** which can be largely improved using MVM. The use of MVM is usually not required after the GA when using the stopping criterion described in **Section 3.4.3**. Here we observe that MVM almost never moves a single node in the partition inferred by the GA-based method. Furthermore, the computation time of gMVM depends on the number of indivisible communities found in the network. Here the maximum number of communities that can be expected is about twenty, which makes the execution of gMVM very fast.

The computation time of Newman's spectral algorithm is constant for the different μ values as we already observed in **Section C.1.3** where we evaluated the computational cost of finding the leading eigenvector v_1 of **B**. Once again, we note a slight increase in the computation time for $\mu > 0.8$, that is, for non-modular networks. This is because the power method takes more iterations to converge when the first and second eigenvectors of **B** are close to each other (**Section C.1.3**). Even if the method proposed by Newman does not have a parallel implementation, Jmod allows to detect the community structure of multiple network simultaneously.

Six threads were used to speed up as many times the GA-based method. Note that its computation time can be even further decreased when more processors are available (e.g. with a cluster). The computation time of the GA-based method increases as the modularity of the network decreases. This results from the fact that communities are less clearly defined

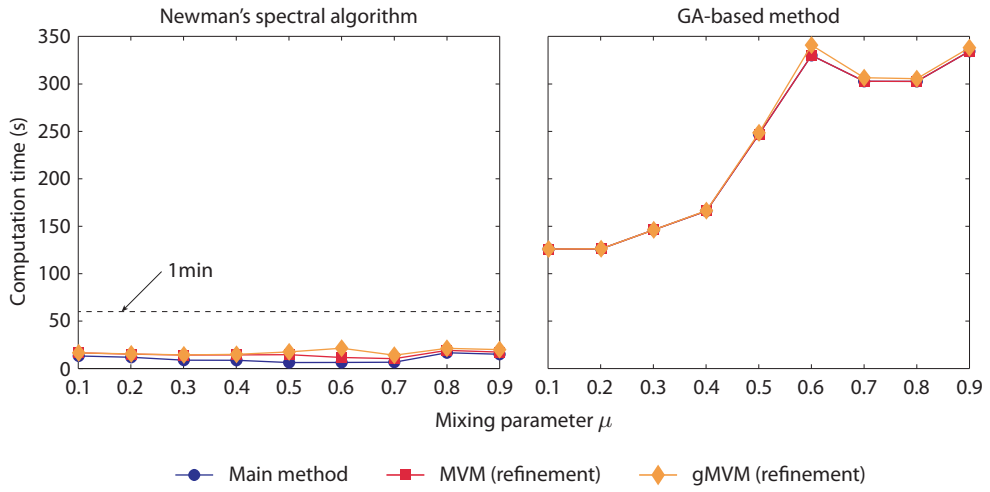


Figure C.3: Computation time of the improved version of Newman's algorithm and GA-based method on 1000-node LFR graphs. These two methods are applied to identify the community structures of 1000-node LFR graphs generated for different values of the mixing parameter μ and to include big communities (20-100 nodes). Newman's spectral algorithm and the GA-based method are successively refined using MVM and gMVM, which only takes a very small amount of time. Newman's algorithm is faster but has lower accuracy than the GA-based method. Moreover, the running time of the latter method depends on the modularity of the network. Low values of μ are used to generate highly modular graphs. Each point is the mean computation time computed for 20 runs of the methods on different graphs.

in non-modular networks, thus the search space of the fitness function Q would look more *flat*. As a consequence, the evaluation of more individuals are required in order to converge towards the optimal solutions. We also observed that the GA stopping criterion described in **Section 3.4.3** requires more generations (proportionally to the size of the network) to be met in larger but similar networks (same node degrees and community sizes). Moreover, the computation time stops to increase after $\mu \approx 0.6$. For these networks, the detection of their community structure becomes difficult and the accuracy of the module inference starts to decrease (**Fig. 3.10**). For $\mu > 0.7$, the method can not capture any more the weak community structures of the networks and always detects on average the same number of communities (**Fig. 3.13**).

Finally, it is interesting to note that the computation time of the GA-based method actually provides information about the modularity of the network for $\mu \in [0.1, 0.6]$, which may be possible used *online* to adapt the behavior of the current community structure detection.

C.4.2 Brute force method

The brute force method introduced in **Section 3.2.3** systematically evaluates the 2^{n-1} split vectors **s** which describe the different ways to split a community in two subcommunities. Here we evaluate its computation time in order to give an idea of the range of its application.

The main limitation of the brute force method is that it can only be applied to split relatively small communities. Indeed, the computation time required doubles each time one node is added to the network. As an example, the running time to split a 30-node network in two is about 37.54 minutes.[c] For Zachary's karate club network[79] and its 34 nodes, the method takes about 644.25 minutes which is about 2^4 times larger than the computation time required for the 30-node network. Moreover, the partitioning of a 50-node graph using the brute force approach would take *more than 80 years* to be completed.

The above computation times correspond to the time it takes to evaluate sequentially every split vectors **s**. In Jmod, the method as been parallelized so that it can use all or a specified number of available processors (each split vector is independent). This actually does not help much as the running time remains proportional to $\frac{2^{n-1}}{T}$ where T is the number of processors or more precisely the number of *threads* that the method can run in parallel. Nevertheless, its main purpose is not to be used in real applications but to provide a ground truth to bi-partitioning methods. However, we also propose to use the brute force method in exhaustive optimization methods to partition small communities that it requires less evaluations than main detection method.

C.5 Detection of the Snap/SNARE in DPiM

Guruharsha *et al.* identified and manually labelled 31 proteins as part of the Snap/SNARE complex of the *Drosophila* protein interaction map (DPiM)[21], which the GA-based module detection method successfully identified as being part of the same community. The name of the 31 proteins as well as their FlyBase[d] identifier are AttD (FBgn0038530), Bet1 (FBgn0260857), CG1599 (FBgn0033452), CG2023 (FBgn0037383), CG6208 (FBgn0037789), gammaSnap (FBgn0028552), Gos28 (FBgn0044871), koko (FBgn0051232), membrin (FBgn0260856), Nsf2 (FBgn0013998), n-syb (FBgn0013342), Rme-8 (FBgn0015477), Sec22 (FBgn0260855), Slh (FBgn0015816), Snap (FBgn0250791), Snap25 (FBgn0011288), Syb (FBgn0003660), Syx1A (FBgn0013343), Syx4 (FBgn0024980), Syx5 (FBgn0011708), Syx7 (FBgn0086377), Syx8 (FBgn0036643), Syx13 (FBgn0036341), Syx16 (FBgn0031106), Syx17 (FBgn0035540), Syx18 (FBgn0039212), Snap24 (FBgn0028401), Use1 (FBgn0035965), usnp (FBgn0034913), Vti1 (FBgn0260862), Ykt6 (FBgn0260858). Furthermore, the method suggests that two additional proteins CG7133 (FBgn0037150) and Sgt (FBgn0032640)[100] may also participate to the Snap/SNARE complex.

[c] Intel® Xeon® CPU X5472 @ 3.00GHz
[d] flybase.org

D Supplementary notes for Chapter 4

D.1 Generation of quantitative datasets

D.1.1 Sample collection

Flies were constantly kept in a 26°C incubator and the eggs were collected on grape juice plates. It is known that the females can keep the fertilized eggs for up to 8h, so a freshly laid egg can be anywhere between minutes to 8h old. We circumvented this problem by treating flies with CO2 prior to collection, which is thought to relax the muscles and facilitate the deposition of old eggs. This first collection was discarded and the flies were transferred to a clean collection chamber. Additionally, as sexual dimorphism exerts itself early on, only male larvae were included in our analysis where possible. Indeed, male flies are comparatively smaller than female flies and including both sexes could bias our scaling results during wing imaginal disc growth. Male larvae were positively selected for by the presence of a clear, oval genital disc which is clearly visible starting from 80 hours after egg laying (AEL). We observed that 70 hours AEL corresponds to the beginning of the third instar stage at 26°C as hatching larvae were frequently encountered. Dissected larvae were fixed immediately, washed and stored at 4°C. Once all time classes were obtained (usually within 2 days), all samples were processed for antibody staining in parallel using identical solutions.

D.1.2 Immunostainings and image acquisition

Samples were transferred into cold fixative (4% pfa in PBS, pH=7) and fixed for 25 min at room temperature on a rotator. Following extensive washes in PBT (PBS + 0.03% TritonX), the discs were blocked in PBTN (PBT + 2% Normal Donkey Serum, Jackson Immuno Research Laboratories) for 1h at 4°C on a rotator, and incubated with primary antibodies overnight at 4°C. The discs were washed several times with cold PBT and incubated in secondary antibodies for 2h at room temperature on a rotator. After another round of washes with PBT, the excess fluid was removed and replaced with Vectashield mounting media (Vector Labs). All discs

from a data set were mounted on the same slide to reduce potential variation in thickness between the slide and the coverslip across different samples. Brain discs were used as spacers. All discs from a dataset were imaged under identical microscopy settings using a Leica SP5 confocal microscope.[a]

Figure 1.2 shows wild type and *pent*$^{2\text{-}5}$ wing discs aged between 80 and 110 hours AEL. The image stacks were acquired using a pixel resolution of 1024 x 1024 and an optical section thickness of 1 µm. The scale of the wings reported in this document is defined as one pixel corresponding to 0.378 µm. This value is consistent throughout unless stated otherwise.

D.1.3 Antibodies and dad-GFP

Rb-α-P-Mad (1:1500, Ed Laufer[199,200]); rb-α-Sal (1:40, Reinhard Schuh[201]); rb-α-Omb (1:1200, Gert O. Pflugfelder[202]); m-α-Wg (a.k.a. 4D4, 1:120, DSHB, University of Iowa[b]); m-α-Ptc (a.k.a. Apa1, 1:600, DSHB, University of Iowa); gp-α-Brk (1:1000, Gines Morata). All secondary antibodies were used in 1:1000 dilutions and were from the AlexaFluor series of Invitrogen. dad-GFP transgenic flies were described in[203].

D.1.4 Preparation for image processing

After the image acquisition, we used the open-source image processing toolkit we developed called WingJ to detect automatically the morphology or structure of the *Drosophila* wing pouch (**Section 4.2**) and quantify gene expression inside it (**Section 4.3**). We also used a tool for segmenting and detecting cell nuclei which is introduced in **Section 4.5**. Different input images were presented depending on the detection tool applied.

Detection of the pouch structure

Wingless (Wg) antibody labelling was used to visualize the outer boundary (or contour) of the pouch and the dorsal/ventral (D/V) compartment boundary while labelling the expression of Patch (Ptc) to delimit the anterior/posterior (A/P) compartment boundary. We removed manually the image slices corresponding to the peripodial membrane from the each Wg-Ptc-AB image stacks. This can be done directly in WingJ by setting the first and last slice index to consider. The detection was then performed on the maximum intensity projection of the remaining image slices.

[a] leica-microsystems.com
[b] dshb.biology.uiowa.edu

Expression quantification

We manually selected by visual inspection five consecutive slices above and below the brightest slice from each stack and performed a mean projection of these eleven slices. Using a reduced number of slices and performing the mean projection allowed us to reduce the noise as well as avoid the signal from the peripodial membrane. Indeed, we made sure that these eleven slices contained signal from the columnar cells of the pouch only.

Detection of cell nuclei

Cell nuclei were stained with TO-PRO-3.[c] We then removed manually the image slices corresponding to the peripodial membrane. For each wing, the remaining part of the TO-PRO-3 image stack was directly given as input to the 3D nuclei detection tool.

D.2 Additional information about the spline-based snake

Proposition 1 *The control points* $\mathbf{c}^{\mathrm{p}}$ *can be obtained from the control points* $\mathbf{c}$ *through the following matrix multiplication.*

$$\begin{pmatrix} \mathbf{c}^{\mathrm{p}}[0]^{\mathrm{T}} \\ \vdots \\ \mathbf{c}^{\mathrm{p}}[M-1]^{\mathrm{T}} \end{pmatrix} = \mathbf{P}_{\mathrm{ref}} \begin{pmatrix} \mathbf{c}[0]^{\mathrm{T}} \\ \vdots \\ \mathbf{c}[M-1]^{\mathrm{T}} \end{pmatrix} \tag{D.1}$$

where the $M \times M$ *matrix* $\mathbf{P}_{\mathrm{ref}}$ *is defined as the product*

$$\mathbf{P}_{\mathrm{ref}} \stackrel{\mathrm{def}}{=} \mathbf{M}_{\mathrm{ref}}^{\mathrm{T}} \left(\mathbf{M}_{\mathrm{ref}} \mathbf{M}_{\mathrm{ref}}^{\mathrm{T}}\right)^{-1} \mathbf{M}_{\mathrm{ref}} \tag{D.2}$$

where

$$\mathbf{M}_{\mathrm{ref}} = \begin{pmatrix} c_x^{\mathrm{ref}}[0] & \dots & c_x^{\mathrm{ref}}[M-1] \\ c_y^{\mathrm{ref}}[0] & \dots & c_y^{\mathrm{ref}}[M-1] \\ 1 & \dots & 1 \end{pmatrix} \tag{D.3}$$

Note that the matrix $\mathbf{P}_{\mathrm{ref}}$ can be precomputed since it only depends on the coefficients of the reference shape. This contributes to significantly speedup the algorithm using lookup tables. Moreover, the use of homogeneous coordinates within the derivation of $\mathbf{P}_{\mathrm{ref}}$ makes possible to represent both the linear transformation and the translation vector as a single multiplication with the matrix

[c] products.invitrogen.com

$$\tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & 1 \end{pmatrix} \tag{D.4}$$

which can also be written as

$$\tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{c}[0] & \dots & \mathbf{c}[M-1] \\ 1 & \dots & 1 \end{pmatrix} \mathbf{M}_{\text{ref}}^{\text{T}} (\mathbf{M}_{\text{ref}} \mathbf{M}_{\text{ref}}^{\text{T}})^{-1} \tag{D.5}$$

Finally, the distance between $\mathbf{r}$ and its elliptical fit simplifies to

$$E_{\text{shape}} = \sum_{k=0}^{M-1} \|\mathbf{c}^{\text{p}}[k] - \mathbf{c}[k]\| \tag{D.6}$$

In our application, the snakes are initialized with a circle shape of radius R whose center is given by the center of gravity of one of the four outer (shaded) triangles of the kite snake (**Fig. 4.8**). The control points of each spline-based snake are initially placed at

$$\mathbf{c}[k] = R \frac{2\left(1-\cos\frac{2\pi}{M}\right)}{\cos\frac{\pi}{M} - \cos\frac{3\pi}{M}} \begin{pmatrix} \cos\frac{2\pi k}{M} \\ \sin\frac{2\pi k}{M} \end{pmatrix} + \mathbf{g}_i \tag{D.7}$$

where each $\mathbf{g}_i, i = 1 \dots 4$ corresponds to the centroid of one of the four outer triangles of the kite snake. The initial radius R is chosen so that there is no overlap among the four initial configurations of the spline-based snakes. We show in **Figure 4.11** the configuration of the four spline-based snakes before and after optimization. Note that the unconstrained optimization algorithm we defined in **Section 4.2.4** for the kite snake is also applied here to minimize the energy function E_{snake} of the spline-based snakes defined in (4.11).

D.3 Unsupervised segmentation of the *Drosophila* embryo

Because the Drosophila embryo is a largely-studied system model, we also implemented a method for enabling the automatic detection and segmentation of the embryo structure. This method reuses some of the image-processing detection modules that we have developed previously to identify the structure of the *Drosophila* wing pouch.

Here, the detection modules are: 1) detection of the contour of the embryo (using active contours), 2) detection of the dorsal-ventral and anterior-posterior axes, 3) automatic inference of the embryo orientation and 4) possibility to manually edit the inferred structure model. We show below two examples of embryo structure quantification using the method that we implemented in WingJ.

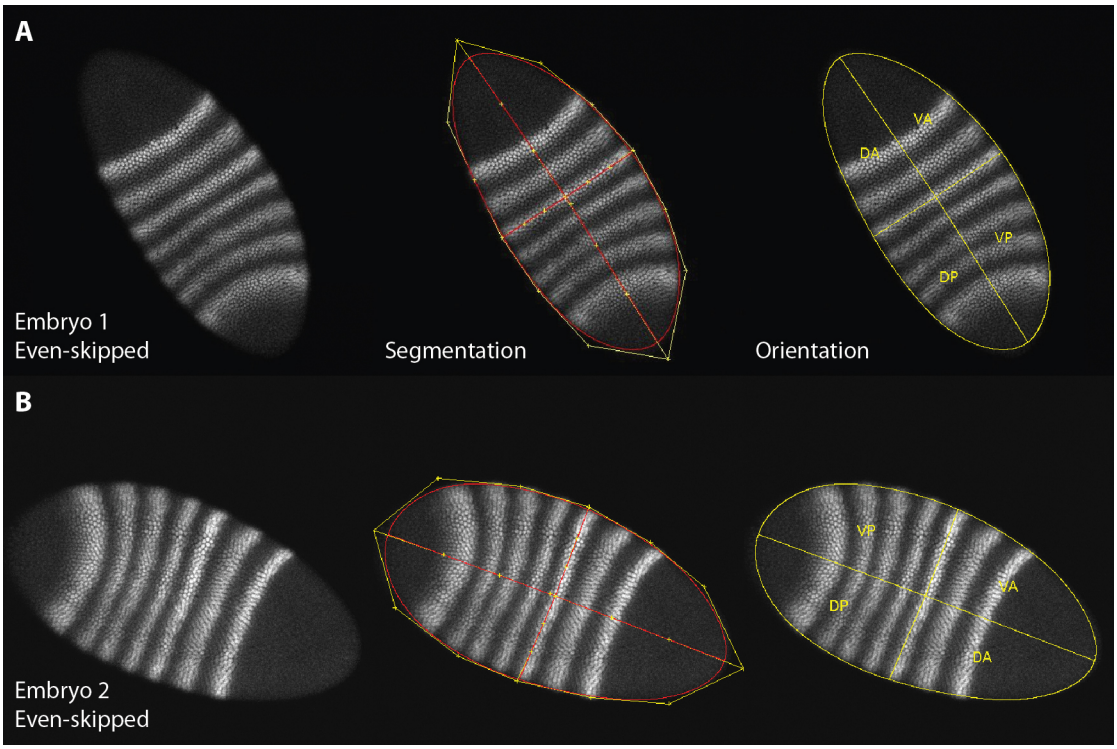


Figure D.1: Unsupervised detection and segmentation of the *Drosophila* embryo. The detection of the *Drosophila* embryo is relatively simple compared to the detection of the *Drosophila* wing pouch (**Section 4.2**). The contour of the embryo is segmented using a snake model similar to the one that we have developed to identify the contour of the wing pouch compartment **Section 4.2.5**. The main difference is that the initial size of the snake covers almost the entire image and contracts itself to fit the outer boundary of the embryo. After manual fine-tuning, the orientation of the embryo structure model is inferred based on the curvature of the A/P and D/V boundary and the non-symmetric expression of a given gene (e.g. *even-skipped*).

E Numerical integration of SDEs

E.1 Introduction

E.1.1 Itô and Stratonovich SDEs

One-dimensional stochastic differentiable equation (SDE) is given by[204,205]

$$\frac{dX_t}{dt} = f(X_t, t)dt + g(X_t, t)dW_t \tag{E.1}$$

where $X_t = X(t)$ is the realization of a stochastic process or random variable. $f(X_t, t)$ is called the *drift coefficient*, that is the deterministic part of the SDE characterizing the local trend. $g(X_t, t)$ denotes the *diffusion coefficient*, that is the stochastic part which influences the average size of the fluctuations of X. The fluctuations themselves originate from the stochastic process W_t called *Wiener process* and introduced in Section E.1.2. Interpreted as an integral, we get

$$X_t = X_{t_0} + \int_{t_0}^{t} f(X_s, s)ds + \int_{t_0}^{t} g(X_s, s)dW_s \tag{E.2}$$

where the first integral is an ordinary Riemann integral. As the sample paths of a Wiener process are not differentiable, the Japanese mathematician K. Itô defined in 1940s a new type of integral called *Itô stochastic integral*. In 1960s, the Russian physicist R. L. Stratonovich proposed an other type of stochastic integral called *Stratonovich stochastic integral* and used the symbol "$\circ$" to distinct it from the former Itô integral. (E.3) and (E.4) are the Stratonovich equivalents of (E.1) and (E.2)[204,206].

$$\frac{dX_t}{dt} = f(X_t, t)dt + g(X_t, t) \circ dW_t \tag{E.3}$$

$$X_t = X_{t_0} + \int_{t_0}^{t} f(X_s, s)ds + \int_{t_0}^{t} g(X_s, s) \circ dW_s \tag{E.4}$$

The second integral in (E.2) and (E.4) can be written in a general form as[207]

$$\int_{t_0}^{t} g(X_s, s) dW_s = \lim_{h \to 0} \sum_{k=0}^{m-1} g(X_{\tau_k}, \tau_k)(W(t_{k+1}) - W(t_k)) \tag{E.5}$$

where $h = (t_{k+1} - t_k)$ with intermediary points $\tau_k = (1-\lambda)t_k - \lambda t_{k+1}$, $\forall k \in \{0, 1, ..., m-1\}$, $\lambda \in [0,1]$. In the stochastic integral of the Itô SDE given in (E.2), $\lambda = 0$ leads to $\tau_k = t_k$ and hence to evaluate the stochastic integral at the left-point of the intervals. In the definition of the Stratonovich integral, $\lambda = 1/2$ and so $\tau_k = (t_{k+1} - t_k)/2$, what fixes the evaluations of the second integral in (E.4) at the mid-point of each interval[207].

To illustrate the difference between the Itô and Stratonovich calculi, we have a closer look at the stochastic integral

$$\int_{t_0}^{T} W(s) dW_s = \lim_{m \to \infty} \sum_{k=0}^{m-1} W(\tau_k)(W(t_{k+1}) - W(t_k)) \tag{E.6}$$

$$= \frac{W(t)}{2} + (\lambda - \frac{1}{2})T \tag{E.7}$$

By combining the result of (E.7) with the respective values of λ discussed above for both interpretations, we obtain[207]

$$\int_{t_0}^{T} W(s) dW_s = \frac{1}{2} W(t) - \frac{1}{2} T \tag{E.8}$$

$$\int_{t_0}^{T} W(s) \circ dW_s = \frac{1}{2} W(t) \tag{E.9}$$

If we solve (E.2) and (E.4) whose stochastic integrals (E.8) and (E.9) are respectively part of, we see that the Itô and Stratonovich representations *do not converge towards the same solution*. Conversions from Itô to Stratonovich calculus and inversely are possible in order to switch between the two different calculi. This is achieved by adding a correction term to the drift coefficients[206].

$$dX_t = f(X_t) dt + g(X_t) dW_t \tag{E.10}$$

$$dX_t = \underline{f}(X_t) dt + g(X_t) \circ dW_t \tag{E.11}$$

$$\underline{f} = f - \frac{1}{2} g' g \tag{E.12}$$

where $g' = \frac{dg(X_t)}{dX_t}$ is the first derivative of g. If the relation (E.12) is used (called the *Itô-Stratonovich drift correction formula*), the integration of the Stratonovich SDE (E.11) leads now to the same result as the integration of the Itô SDE (E.10)[206].

Both integrals have their advantages and disadvantages and which one should be used is more a modelling than mathematical issue. In financial mathematics, the Itô interpretation is usually used since Itô calculus only takes into account information about the past. The Stratonovich interpretation is the most frequently used within the physical sciences[204]. An excellent discussion of this subject can be found in [208], in particular see Chapter IX, Section 5: *The Itô-Stratonovich dilemma.*

E.1.2 Standard Wiener process

A scalar *standard Brownian motion,* or *standard Wiener process,* over $[t_0, T]$ is a random variable $W(t)$ that depends continuously on $t \in [t_0, T]$. For $t_0 \leqslant s < t \leqslant T$, the random variable given by the increment $W(t) - W(s)$ is normally distributed with mean $\mu = 0$ and variance $\sigma^2 = t - s$. Equivalently, $W(t) - W(s) \sim \sqrt{t-s}\mathcal{N}(0,1)$ with $W(t_0 = 0) = 0$. The conditions for the stochastic process $W(t)$ to be a Wiener process are[204,209]

1. $[W(t), t \geqslant 0]$ has stationary independent increments dW
2. $W(t)$ is normally distributed for $t \geqslant 0$
3. $\langle W(t) \rangle = 0$ for $t \geqslant 0$
4. $W(0) = 0$

E.1.3 Discretized Brownian motion

We take $t_0 = 0$ and divide the interval $[0, T]$ into N steps such as: $h = T/N$. We also denote $W_j = W(t_j)$ with $t_j = jh$[209].

$$W_j = W_{j-1} + dW_j \qquad W_0 = 0 \qquad j = 1, 2, ..., N \tag{E.13}$$

where each dW_j is an independent random variable of the form $\sqrt{h}\mathcal{N}(0,1)$.[a] **Figure E.1** shows the realizations of three independent Wiener processes.

E.2 Numerical integration

E.2.1 Iterative methods

It is difficult to deal with the SDEs analytically because of the highly non-differentiable character of the realization of the Wiener process. There are different, iterative methods that can be used to integrate SDE systems. The most widely-used ones are introduced in the following sections.

[a] $\mathcal{N}(0,h) = \sqrt{h}\mathcal{N}(0,1)$

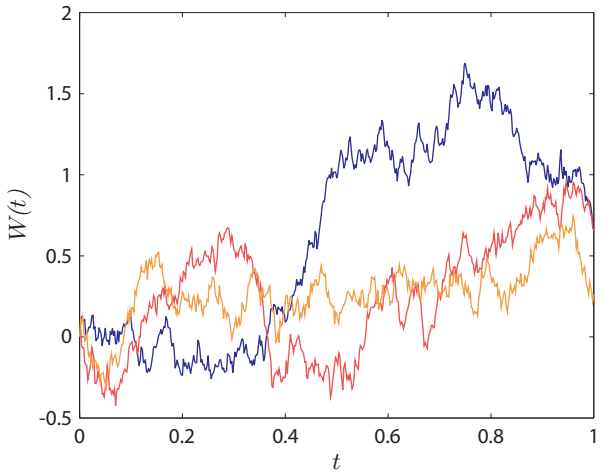


Figure E.1: Simulation of one-dimensional Brownian paths. Three discretized, one-dimensional Brownian paths with $T = 1$ and $N = 500$. When $t \to \infty$, the process has an infinite variance but still an expectation equal to zero.

- **Explicit order 0.5 strong Taylor scheme**
  Euler-Maruyama (EM) and Euler-Heun (EH)
- **Explicit order 1.0 strong Taylor scheme**
  Milstein and derivative-free Milstein (Runge-Kutta approach)
- **Explicit order 1.5 strong Taylor scheme**
  Stochastic Runge-Kutta (SRK)

E.2.2 Explicit order 0.5 strong Taylor scheme

Euler-Maruyama method

The simplest stochastic numerical approximation is the Euler-Maruyama method and requires the problem to be described using the Itô scheme. For Stratonovich interpretation, use the Euler-Heun method described in Section E.2.2.

This approximation is a continuous time stochastic process that satisfy the iterative scheme[210]

$$Y_{n+1} = Y_n + f(Y_n)h_n + g(Y_n)\Delta W_n \qquad Y_0 = x_0 \qquad n = 0, 1, ..., N-1 \tag{E.14}$$

$$\Delta W_n = [W_{t+h} - W_t] \sim \sqrt{h}\mathcal{N}(0,1) \tag{E.15}$$

where $Y_n = Y(t_n)$, $h_n = t_{n+1} - t_n$ is the step size, $\Delta W_n = W(t_{n+1}) - W(t_n) \sim \mathcal{N}(0, h_n)$ with $W(t_0) = 0$. From now on, we use the following notation: $h = h_n$ (fixed step size), $f_n = f(Y_n)$

and $g_n = g(Y_n)$. (E.14) becomes

$$Y_{n+1} = Y_n + f_n h + g_n \Delta W_n \tag{E.16}$$

As the order of convergence for the Euler-Maruyama method is low (strong order of convergence 0.5, weak order of convergence 1), the numerical results are inaccurate unless a small step size is used. In fact, Euler-Maruyama represents the *order 0.5 strong Taylor scheme*. By adding one more term from the stochastic Taylor expansion, one obtains a 1.0 strong order of convergence scheme known as *Milstein scheme*[210].

Euler-Heun method

If a problem is described using the Stratonovich scheme, then the Euler-Heun method has to be used instead of the Euler-Maruyama method which is only valid for Itô SDEs[204,211].

$$Y_{n+1} = Y_n + f_n h + \frac{1}{2}\left[g_n + g(\bar{Y}_n)\right]\Delta W_n \tag{E.17}$$

$$\bar{Y}_n = Y_n + g_n \Delta W_n \tag{E.18}$$

$$\Delta W_n = [W_{t+h} - W_t] \sim \sqrt{h}\mathcal{N}(0,1) \tag{E.19}$$

E.2.3 Explicit order 1.0 strong Taylor scheme

Milstein method

The Milstein scheme is slightly different for the Itô and Stratonovich representations[204,205,211]. It can be proved that Milstein scheme converges strongly with order 1 (and weakly with order 1) to the solution of the SDE. The Milstein scheme represents the *order 1.0 strong Taylor scheme*.

$$Y_{n+1} = Y_n + f_n h + g_n \Delta W_n + \frac{1}{2} g_n g_n' \left[(\Delta W_n)^2 - h\right] \tag{E.20}$$

$$Y_{n+1} = Y_n + f_n h + g_n \Delta W_n + \frac{1}{2} g_n g_n' (\Delta W_n)^2 \tag{E.21}$$

$$\Delta W_n = [W_{t+h} - W_t] \sim \sqrt{h}\mathcal{N}(0,1) \tag{E.22}$$

where $g_n' = \frac{dg(Y_n)}{dY_n}$ is the first derivative of g_n. The iterative method defined by (E.20) must be used with Itô SDEs and (E.21) with Stratonovich SDEs. Note that when *additive noise* is used, i.e. when g_n is constant and so independent of Y_n, then both Itô and Stratonovich interpretations are equivalent ($g_n' = 0$).

Derivative-free Milstein method

The drawback of the previous method is that it requires the analytic specification of the first derivative of $g(Y_n)$, analytic expression which can become quickly highly complex. The following implementation approximates this derivative using a Runge-Kutta approach[204].

$$Y_{n+1} = Y_n + f_n h + g_n \Delta W_n + \frac{1}{2\sqrt{h}} \left[g(\bar{Y}_n) - g_n \right] \left[(\Delta W_n)^2 - h \right] \tag{E.23}$$

$$Y_{n+1} = Y_n + f_n h + g_n \Delta W_n + \frac{1}{2\sqrt{h}} \left[g(\bar{Y}_n) - g_n \right] (\Delta W_n)^2 \tag{E.24}$$

$$\bar{Y}_n = Y_n + f_n h + g_n \sqrt{h} \tag{E.25}$$

$$\Delta W_n = [W_{t+h} - W_t] \sim \sqrt{h} \mathcal{N}(0,1) \tag{E.26}$$

where (E.23) and (E.24) must be applied respectively to Itô and Stratonovich SDEs.

E.2.4 Explicit order 1.5 strong Taylor scheme

Definition

By adding more terms from the stochastic Taylor expansion than in Milstein scheme, higher strong orders can be obtained. A method to generate a strong order 1.5 method is introduced by Burrage & Platen[212,213]. For the need of this method, a random variable ΔZ_n is introduced.

$$\Delta Z_n = \int_{\tau_n}^{\tau_{n+1}} \int_{\tau_n}^{\tau_{s_2}} dW_{s_1} ds_2 \tag{E.27}$$

which is a Gaussian distributed with mean zero, variance $\frac{1}{3}h^3$ and correlation $E(\Delta W_n \Delta Z_n) = \frac{1}{2}h^2$.

Stochastic Runge-Kutta

This implementation allows to achieve a 1.5 strong order of converge. This is the highest strong order obtained with a Runge-Kutta approach that keeps a "simple" structure. This implementation makes use of the ΔZ_n introduced in (E.27)[212,213].

$$\Delta Y_{n+1} = Y_n + f_n h + g_n \Delta W_n + \frac{1}{2} g_n g_n' \left[(\Delta W_n)^2 - h \right] \tag{E.28}$$

$$+ f_n' g_n \Delta Z_n + \frac{1}{2} \left[f_n f_n' + \frac{1}{2} g_n^2 f_n'' \right] h^2 \tag{E.29}$$

$$+ \left[f_n g_n' + \frac{1}{2} g_n^2 g_n'' \right] [\Delta W_n h - \Delta Z_n] \tag{E.30}$$

$$+\frac{1}{2}g_n\left[g_n g_n'' + (g_n')^2\right]\left[\frac{1}{3}(\Delta W_n)^2 - h\right]\Delta W_n \tag{E.31}$$

E.3 Convergence

An approximation Y *converges with strong order* $\gamma > 0$ if there exists a constant K such that

$$E\left(|X_T - Y_N|\right) \leqslant K \cdot h^{\gamma} \tag{E.32}$$

for step sizes $h \in (0,1)$, with X_T being the true solution at time T and Y_N the approximation[212]. The symbol E stands for *expectation*. It appears that Euler-Maruyama scheme converges only with strong order $\gamma = 0.5$. Strong approximation is tightly linked to the use of the original increments of the Wiener process[212]. However in several applications, it is not needed to simulate a pathwise approximation of a Wiener process. For instance, one could be only interested in the moments of the solution of a SDE. A discrete time approximation Y *converges with weak order* $\beta > 0$ if for any polynomial $g(\cdot)$ there exists a constant K_g such that

$$|E(g(X_T)) - E(g(Y_N))| \leqslant K_g \cdot h^{\beta} \tag{E.33}$$

for step sizes $h \in (0,1)$. It turns out that Euler-Maruyama scheme converges with weak order $\beta = 1.0$.

If a numerical method is convergent with order γ and the step size is made k times smaller, then the approximation error decreases by a factor k^{γ}. For instance, if the order is equal to 1 and we want to decrease the error 100 times, we have to make the step size 100 times smaller. If the order is equal to 0.5 and we still want to decrease the error 100 times, we have to make the step size $100^2 = 10000$ times smaller.

E.4 libSDE: Java library for simulating SDEs

E.4.1 Overview

Stochastic differential equations were initially used in this work to model and simulate molecular noise in the dynamics of gene regulatory networks generated using GNW (**Section 2.2.2**). Implementing one integration method such as the derivative-free Mistein method was all we required. However because there was no other Java library available at the time of this project, I decided to invest more time to develop an extensible Java library implementing all the numerical integration methods described above.

libSDE is released under open-source license and can be downloaded from its project webpage (tschaffter.ch/projects/libsde). Here is a list of possible actions that users can do using libSDE:

- Simulate Itô and Stratonovich SDEs
- Select an integration method among *Euler-Maruyama, Euler-Heun, derivative-free Milstein*, and *Stochastic Runge-Kutta (SRK15) solvers*
- Set the integration step-size t
- Set the time interval $[t_0, T]$
- Set the number of trajectories (time series) to simulate
- Set the number of time points per time series
- Implement additional integration methods (libSDE provides a factory method pattern)
- Integrate libSDE to third-party computational tools

E.4.2 Example

We generate stochastic simulations of $dX = (-3X + 1)dt + \sigma dW$ with $X(0) = 1$ and $\sigma = 0.2$ (**Fig. E.2**). The expected solution $E(X)$ is equal to $2/3e^{-3t} + 1/3$ (in orange). Here Ito and Stratonovich schemes are equivalent because the drift coefficient is a constant term. dX is integrated using the SRK15 solver implemented in libSDE.

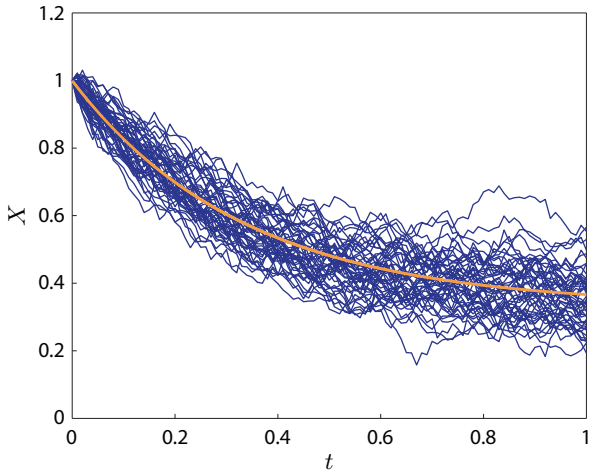


Figure E.2: Example of numerical integration using libSDE. The example code included in libSDE integrates fifty times the stochastic equation $dX = (-3X + 1)dt + \sigma dW$ with $X(0) = 1$ and $\sigma = 0.2$ using the derivative-free Milstein method (in blue). The expected solution is $E(X) = 2/3e^{-3t} + 1/3$ (in orange). Note that here the Itô and Stratonovich drift terms are equal because sigma is constant.

Bibliography

1. Davis, N., Favero, T., de Hoog, ., *et al.* Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science* **270,** 467 (1995).
2. Lockhart, D. *et al.* Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology* **14,** 1675 (1996).
3. Mortazavi, A., Williams, B., McCue, K., Schaeffer, L. & Wold, B. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature methods* **5,** 621–628 (2008).
4. Marbach, D. *et al.* Wisdom of crowds for robust gene network inference. *Nature Methods* (2012).
5. Kim, S., Imoto, S. & Miyano, S. Inferring gene networks from time series microarray data using dynamic Bayesian networks. *Briefings in Bioinformatics* **4,** 228. ISSN: 1467-5463 (2003).
6. Gama-Castro, S. *et al.* RegulonDB version 7.0: transcriptional regulation of Escherichia coli K-12 integrated within genetic sensory response units (Gensor Units). *Nucleic Acids Research* **39,** D98. ISSN: 0305-1048 (2011).
7. Camacho, D. & Collins, J. Systems biology strikes gold. *Cell* **137,** 24. ISSN: 1097-4172 (2009).
8. Cantone, I. *et al.* A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. *Cell* **137,** 172–181. ISSN: 0092-8674 (2009).
9. Kremling, A. *et al.* A benchmark for methods in reverse engineering and model discrimination: problem formulation and solutions. *Genome research* **14,** 1773. ISSN: 1088-9051 (2004).
10. Mendes, P., Sha, W. & Ye, K. Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics* **19.** ISSN: 1367-4803 (2003).
11. Den Bulcke, T. V. *et al.* SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC bioinformatics* **7,** 43. ISSN: 1471-2105 (2006).
12. Camillo, B. D., Toffolo, G. & Cobelli, C. A gene network simulator to assess reverse engineering algorithms. *Annals of the New York Academy of Sciences* **1158,** 125–142. ISSN: 1749-6632 (2009).

13. Ravasz, E., Somera, A., Mongru, D., Oltvai, Z. & Barabási, A. Hierarchical organization of modularity in metabolic networks. *Science* **297,** 1551 (2002).

14. Shen-Orr, S., Milo, R., Mangan, S. & Alon, U. Network motifs in the transcriptional regulation network of Escherichia coli. *Nature genetics* **31,** 64–68 (2002).

15. Li, Y. *et al.* ReTRN: A retriever of real transcriptional regulatory network and expression data for evaluating structure learning algorithm. *Genomics* **94,** 349–354. ISSN: 0888-7543 (2009).

16. Roy, S., Werner-Washburne, M. & Lane, T. A system for generating transcription regulatory networks with combinatorial control of transcription. *Bioinformatics* **24,** 1318. ISSN: 1367-4803 (2008).

17. Hache, H., Wierling, C., Lehrach, H. & Herwig, R. GeNGe: systematic generation of gene regulatory networks. *Bioinformatics* **25,** 1205. ISSN: 1367-4803 (2009).

18. Belle, A., Tanay, A., Bitincka, L., Shamir, R. & O'Shea, E. Quantification of protein half-lives in the budding yeast proteome. *Proceedings of the National Academy of Sciences* **103,** 13004 (2006).

19. Haynes, B. & Brent, M. Benchmarking regulatory network reconstruction with GRENDEL. *Bioinformatics* **25,** 801. ISSN: 1367-4803 (2009).

20. Watts, D., Dodds, P. & Newman, M. Identity and search in social networks. *Science* **296,** 1302 (2002).

21. Guruharsha, K. *et al.* A Protein Complex Network of Drosophila melanogaster. *Cell* **147,** 690–703 (2011).

22. Bullmore, E. & Sporns, O. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience* **10,** 186–198 (2009).

23. Krackhardt, D. Cognitive social structures. *Social Networks* **9,** 109–134 (1987).

24. Newman, M. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems* **38,** 321–330 (2004).

25. Lancichinetti, A. & Fortunato, S. Community detection algorithms: a comparative analysis. *Physical Review E* **80,** 056117 (2009).

26. Girvan, M. & Newman, M. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* **99,** 7821 (2002).

27. Newman, M. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* **103,** 8577 (2006).

28. Newman, M. E. Finding community structure in networks using the eigenvectors of matrices. *Physical review E* **74,** 036104 (2006).

29. Shen, H.-W. & Cheng, X.-Q. Spectral methods for the detection of network community structure: a comparative analysis. *Journal of Statistical Mechanics: Theory and Experiment* **2010,** P10020 (2010).

30. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V. & Parisi, D. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America* **101,** 2658 (2004).

31. Clauset, A., Newman, M. & Moore, C. Finding community structure in very large networks. *Physical review E* **70,** 066111 (2004).

32. Guimera, R. & Amaral, L. Functional cartography of complex metabolic networks. *Nature* **433,** 895–900 (2005).

33. Medus, A., Acuna, G. & Dorso, C. Detection of community structures in networks via global optimization. *Physica A: Statistical Mechanics and its Applications* **358,** 593–604 (2005).

34. Massen, C. & Doye, J. Identifying communities within energy landscapes. *Physical Review E* **71,** 046101 (2005).

35. Rosvall, M. & Bergstrom, C. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences* **105,** 1118 (2008).

36. Shen, H. *et al.* Automatic tracking of biological cells and compartments using particle filters and active contours. *Chemometrics and Intelligent Laboratory Systems* **82,** 276–282 (2006).

37. Meijering, E., Dzyubachyk, O., Smal, I. & van Cappellen, W. A. in *Seminars in cell & developmental biology* **20** (2009), 894–902.

38. Delgado-Gonzalo, R., Thevenaz, P., Seelamantula, C. S. & Unser, M. Snakes with an ellipse-reproducing property. *Image Processing, IEEE Transactions on* **21,** 1258–1271 (2012).

39. Crombach, A., Wotton, K. R., Cicin-Sain, D., Ashyraliyev, M. & Jaeger, J. Efficient Reverse-Engineering of a Developmental Gene Regulatory Network. *PLoS Computational Biology* **8,** e1002589 (2012).

40. Jaeger, J. *et al.* Dynamic control of positional information in the early Drosophila embryo. *Nature* **430,** 368–371 (2004).

41. Jaeger, J. The gap gene network. *Cellular and Molecular Life Sciences* **68,** 243–274 (2011).

42. Morata, G. How Drosophila appendages develop. *Nature Reviews Molecular Cell Biology* **2,** 89–97 (2001).

43. Martín, F., Herrera, S. & Morata, G. Cell competition, growth and size control in the Drosophila wing imaginal disc. *Development* **136,** 3747–3756 (2009).

44. Lawrence, P. & Struhl, G. Morphogens, compartments, and pattern: lessons from drosophila? *Cell* **85,** 951 (1996).

45. Affolter, M. & Basler, K. The Decapentaplegic morphogen gradient: from pattern formation to growth regulation. *Nature Reviews Genetics* **8,** 663–674 (2007).

46. García-Bellido, A., Ripoll, P. & Morata, G. Developmental compartmentalisation of the wing disk of Drosophila. *Nature* **245,** 251–253 (1973).

47. Aldaz, S., Escudero, L. M. & Freeman, M. Live imaging of Drosophila imaginal disc development. *Proceedings of the National Academy of Sciences* **107,** 14217–14222 (2010).

48. Wolpert, L. *et al. Principles of development* (Oxford University Press New York, 2002).

49. Neufeld, T. P., de la Cruz, A. F. A., Johnston, L. A. & Edgar, B. A. Coordination of Growth and Cell Division in the *Drosophila* Wing. *Cell* **93,** 1183–1193 (1998).

50. Hamaratoglu, F., de Lachapelle, A., Pyrowolakis, G., Bergmann, S. & Affolter, M. Dpp Signaling Activity Requires Pentagone to Scale with Tissue Size in the Growing Drosophila Wing Imaginal Disc. *PLoS biology* **9,** e1001182 (2011).

51. Schaffter, T., Marbach, D. & Floreano, D. GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics* **27,** 2263–2270 (2011).

52. Balaji, S., Babu, M., Iyer, L., Luscombe, N. & Aravind, L. Comprehensive analysis of combinatorial regulation using the transcriptional regulatory network of yeast. *Journal of molecular biology* **360,** 213–227. ISSN: 0022-2836 (2006).

53. Ackers, G., Johnson, A. & Shea, M. Quantitative model for gene regulation by lambda phage repressor. *Proceedings of the National Academy of Sciences of the United States of America* **79,** 1129 (1982).

54. Stolovitzky, G. *et al.* Statistical analysis of MPSS measurements: application to the study of LPS-activated macrophage gene expression. *Proceedings of the National Academy of Sciences of the United States of America* **102,** 1402 (2005).

55. Marbach, D., Schaffter, T., Mattiussi, C. & Floreano, D. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of Computational Biology* **16,** 229–239 (2009).

56. Rice, J., Tu, Y. & Stolovitzky, G. Reconstructing biological networks using conditional correlation analysis. *Bioinformatics* **21,** 765. ISSN: 1367-4803 (2005).

57. Margolin, A. *et al.* ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC bioinformatics* **7,** S7. ISSN: 1471-2105 (2006).

58. Faith, J. *et al.* Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biol* **5,** e8 (2007).

59. Marbach, D., Mattiussi, C. & Floreano, D. Replaying the evolutionary tape: Biomimetic reverse engineering of gene networks. *Annals of the New York Academy of Sciences* **1158,** 234–245. ISSN: 1749-6632 (2009).

60. Bonneau, R. *et al.* The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome biology* **7,** R36 (2006).

61. Shannon, P. *et al.* Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research* **13,** 2498–2504 (2003).

62. Marbach, D. *et al.* Revealing strengths and weaknesses of methods for gene network inference. *Proceedings of the National Academy of Sciences* **107,** 6286–6291 (2010).

63. Becskei, A. & Serrano, L. Engineering stability in gene networks by autoregulation. *Nature* **405,** 590–593. ISSN: 0028-0836 (2000).

64. Gardner, T. & Collins, J. Neutralizing noise in gene networks. *Nature* **405,** 520–1 (2000).

65. Gillespie, D. The chemical Langevin equation. *The Journal of Chemical Physics* **113,** 297 (2000).

66. Prill, R. *et al.* Towards a rigorous assessment of systems biology models: the DREAM3 challenges. *PloS one* **5,** e9202 (2010).

67. Davis, J. & Goadrich, M. in *Proceedings of the 23rd international conference on Machine learning* (2006), 233–240. ISBN: 1595933832.

68. Klamt, S., Flassig, R. & Sundmacher, K. TRANSWESD: inferring cellular networks with transitive reduction. *Bioinformatics* **26,** 2160. ISSN: 1367-4803 (2010).

69. Menéndez, P., Kourmpetis, Y., Braak, C. T., van Eeuwijk, F. & Isalan, M. Gene Regulatory Networks from Multifactorial Perturbations Using Graphical Lasso: Application to the DREAM4 Challenge. *PloS one* **5,** e14147 (2010).

70. Pinna, A., Soranzo, N. & de la Fuente, A. From Knockouts to Networks: Establishing Direct Cause-Effect Relationships through Graph Analysis. *PloS one* **5,** 218–223. ISSN: 1932-6203 (2010).

71. Yip, K., Alexander, R., Yan, K. & Gerstein, M. Improved reconstruction of in silico gene regulatory networks by integrating knockout and perturbation data. *PloS one* **5** (2010).

72. Fruchterman, T. M. & Reingold, E. M. Graph drawing by force-directed placement. *Software: Practice and experience* **21,** 1129–1164 (1991).

73. Kashtan, N. & Alon, U. Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America* **102,** 13773–13778 (2005).

74. Vijender, C., Preetam, G., Edward, P., Ping, G. & Chaoyang, Z. Time lagged information theoretic approaches to the reverse engineering of gene regulatory networks. *BMC Bioinformatics* **11.** ISSN: 1471-2105 (2010).

75. Danon, L., Diaz-Guilera, A., Duch, J. & Arenas, A. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment* **2005,** P09008 (2005).

76. Fortunato, S. Community detection in graphs. *Physics Reports* **486,** 75–174 (2010).

77. Fortunato, S. & Barthelemy, M. Resolution limit in community detection. *Proceedings of the National Academy of Sciences* **104,** 36 (2007).

78. Newman, M. & Girvan, M. Finding and evaluating community structure in networks. *Physical review E* **69,** 026113 (2004).

79. Zachary, W. An information flow model for conflict and fission in small groups. *Journal of anthropological research,* 452–473 (1977).

80. Goldberg, D. *Genetic algorithms in search, optimization, and machine learning* (Addison-wesley, 1989).

81. Floreano, D. & Mattiussi, C. *Bio-inspired artificial intelligence: theories, methods, and technologies* (The MIT Press, 2008).

82. Holland, J. H. Adaptation in Natural and Artificial Systems. *University of Michigan Press, Ann Arbor* (1975).

83. Goldberg, D. E. & Deb, K. A comparative analysis of selection schemes used in genetic algorithms. *Urbana* **51,** 61801–2996 (1991).

84. Miller, B. L. & Goldberg, D. E. Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation* **4,** 113–131 (1996).

85. Back, T., Fogel, D. B. & Michalewicz, Z. *Handbook of evolutionary computation* (IOP Publishing Ltd., 1997).

86. Duch, J. & Arenas, A. Community detection in complex networks using extremal optimization. *Physical Review E* **72,** 027104 (2005).

87. Kernighan, B. & Lin, S. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal* **49,** 291–307 (1970).

88. Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* **2008,** P10008 (2008).

89. Schuetz, P. & Caflisch, A. Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Physical Review E* **77,** 046112 (2008).

90. Lancichinetti, A., Fortunato, S. & Radicchi, F. Benchmark graphs for testing community detection algorithms. *Physical Review E* **78,** 046110 (2008).

91. Aldecoa, R. & Marín, I. Surprise maximization reveals the community structure of complex networks. *Scientific reports* **3** (2013).

92. Mitchell, M. *An introduction to genetic algorithms* (The MIT press, 1998).

93. Holland, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence* (MIT press, 1992).

94. Morrison, R. W. & De Jong, K. A. in *Artificial Evolution* (2002), 31–41.

95. Mattiussi, C., Waibel, M. & Floreano, D. Measures of diversity for populations and distances between individuals with highly reorganizable genomes. *Evolutionary Computation* **12,** 495–515 (2004).

96. Doreian, P. On the connectivity of social networks†. *Journal of Mathematical Sociology* **3,** 245–258 (1974).

97. Good, B., de Montjoye, Y. & Clauset, A. Performance of modularity maximization in practical contexts. *Physical Review E* **81,** 046106 (2010).

98. Strehl, A. & Ghosh, J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research* **3,** 583–617 (2003).

99. Fred, A. L. & Jain, A. K. Combining multiple clusterings using evidence accumulation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **27,** 835–850 (2005).

100. Gelbart, W. *et al.* The FlyBase database of the Drosophila genome projects and community literature. *Nucleic Acids Research* **27** (1999).

101. Arenas, A., Fernandez, A. & Gomez, S. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics* **10,** 053039 (2008).

102. Lancichinetti, A. & Fortunato, S. Consensus clustering in complex networks. *Scientific reports* **2** (2012).

103. Wallis, J., Miller, T., Lerner, C. & Kleerup, E. Three-dimensional display in nuclear medicine. *IEEE Transactions on Medical Imaging* **8,** 297–230 (1989).

104. Bovik, A. *The Essential Guide to Image Processing* (Academic Press, 2009).

105. Scharf, L. L. & Demeure, C. *Statistical signal processing: detection, estimation, and time series analysis* (Addison-Wesley Publishing Company, 1991).

106. Prewitt, J. & Mendelsohn, M. The Analysis of Cell Images. *Annals of the New York Academy of Sciences* **128,** 1035–1053 (1966).

107. Zhang, T. & Suen, C. A Fast Parallel Algorithm for Thinning Digital Patterns. *Communications of the ACM* **27,** 236–239 (1984).

108. Trier, Ø., Jain, A. & Taxt, T. Feature Extraction Methods for Character Rrecognition - A Survey. *Pattern Recognition* **29,** 641–662 (1996).

109. Abeysinghe, S. S., Baker, M., Chiu, W. & Ju, T. in *Shape Modeling and Applications, 2008. SMI 2008. IEEE International Conference on* (2008), 63–71.

110. Dimitrov, P., Phillips, C. & Siddiqi, K. in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on* **1** (2000), 417–423.

111. Ogniewicz, R. & Kübler, O. Hierarchic Voronoi Skeletons. *Pattern Recognition* **28,** 343–359 (1995).

112. Ju, T., Baker, M. & Chiu, W. Computing a Family of Skeletons of Volumetric Models for Shape Description. *Computer-Aided Design* **39,** 352–360 (2007).

113. Ghosh, R. & Webb, W. Automated Detection and Tracking of Individual and Clustered Cell Surface Low Density Lipoprotein Receptor Molecules. *Biophysical Journal* **66,** 1301–1318 (1994).

114. Harder, N. *et al.* Automated analysis of the mitotic phases of human cells in 3D fluorescence microscopy image sequences. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006,* 840–848 (2006).

115. Wang, M. *et al.* Novel Cell Segmentation and Online SVM for Cell Cycle Phase Identification in Automated Microscopy. *Bioinformatics* **24,** 94–101 (2008).

116. Baron, R. Mechanisms of Human Facial Recognition. *International Journal of Man-Machine Studies* **15,** 137–178 (1981).

117. Brunelli, R. & Poggio, T. Template Matching: Matched Spatial Filters and Beyond. *Pattern Recognition* **30,** 751–768 (1997).

118. Chang, K., Bowyer, K. & Flynn, P. Multiple Nose Region Matching for 3D Face Recognition Under Varying Facial Expression. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28,** 1695–1700 (2006).

119. Soille, P. *Morphological Image Analysis: Principles and Applications* (Springer-Verlag, 2003).

120. Grau, V., Mewes, A. a. A. M., Kikinis, R. & Warfield, S. Improved Watershed Transform for Medical Image Segmentation Using Prior Information. *IEEE Transactions on Medical Imaging* **23,** 447–458 (2004).

121. McInerney, T. & Terzopoulos, D. Deformable Models in Medical Image Analysis: A Survey. *Medical Image Analysis* **1,** 91–108 (1996).

122. Jain, A., Zhong, Y. & Dubuisson-Jolly, M.-P. Deformable Template Models: A Review. *Signal Processing* **71,** 109–129 (1998).

123. Cootes, T., Edwards, G. & Taylor, C. Active Appearance Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23,** 681–685 (2001).

124. Zhu, L., Chen, Y. & Yuille, A. Learning a Hierarchical Deformable Template for Rapid Deformable Object Parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32,** 1029–1043 (2010).

125. Kass, M., Witkin, A. & Terzopoulos, D. Snakes: Active Contour Models. *International Journal of Computer Vision* **1,** 321–331 (1988).

126. Zimmer, C. & Olivo-Marin, J.-C. Coupled Parametric Active Contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27,** 1838–1842 (2005).

127. Thévenaz, P., Delgado-Gonzalo, R. & Unser, M. The Ovuscule. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33,** 382–393 (2011).

128. Staib, L. & Duncan, J. Boundary Finding with Parametrically Deformable Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14,** 1061–1075 (1992).

129. Brigger, P., Hoeg, J. & Unser, M. B-Spline Snakes: A Flexible Tool for Parametric Contour Detection. *IEEE Transactions on Image Processing* **9,** 1484–1496 (2000).

130. Delgado-Gonzalo, R., Thévenaz, P., Seelamantula, C. & Unser, M. Snakes with an Ellipse-Reproducing Property. *IEEE Transactions on Image Processing* **21,** 1258–1271 (2012).

131. Malladi, R., Sethian, J. & Vemuri, B. Shape Modeling with Front Propagation: A Level Set Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17,** 158–175 (1995).

132. Caselles, V., Kimmel, R. & Sapiro, G. Geodesic Active Contours. *International Journal of Computer Vision* **22,** 61–79 (1997).

133. Zhang, K., Zhang, L., Song, H. & Zhou, W. Active Contours with Selective Local or Global Segmentation: A New Formulation and Level Set Method. *Image and Vision Computing* **28,** 668–676 (2010).

134. Jacob, M., Blu, T. & Unser, M. Efficient energies and algorithms for parametric snakes. *Image Processing, IEEE Transactions on* **13,** 1231–1244 (2004).

135. Press, W., Teukolsky, S., Vetterling, W. & Flannery, B. *Numerical Recipes: The Art of Scientific Computing* Third edition, 818 p. (Cambridge University Press, Cambridge, UK, 1986).

136. Delgado-Gonzalo, R., Thévenaz, P. & Unser, M. Exponential splines and minimal-support bases for curve representation. *Computer Aided Geometric Design* **29,** 109–128 (2012).

137. Unser, M. Sampling—50 Years After Shannon. *Proceedings of the IEEE* **88,** 569–587 (2000).

138. Jacob, M., Blu, T. & Unser, M. in *Proceedings of the 2001 SPIE International Symposium on Medical Imaging: Image Processing (MI'01)* **4322** (San Diego, CA, USA, 2001), 340–347.

139. Pavlidis, T. *Algorithms for Graphics and Image Processing* 416 (Computer Science Press, 1982).

140. Melkman, A. On-line Construction of the Convex Hull of a Simple Polyline. *Information Processing Letters* **25,** 11–12 (1987).

141. Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E. & Yergeau, F. Extensible markup language (XML). *World Wide Web Journal* **2,** 27–66 (1997).

142. Waters, J. & Swedlow, J. Interpreting fluorescence microscopy images and measurements. *Cell Online* (2008).

143. Waters, J. Accuracy and precision in quantitative fluorescence microscopy. *The Journal of cell biology* **185,** 1135–1148 (2009).

144. Gerencser, A., Adam-Vizi, V., *et al.* Selective, high-resolution fluorescence imaging of mitochondrial Ca2+ concentration. *Cell Calcium* **30,** 311 (2001).

145. Lachapelle, A. D. & Bergmann, S. Precision and scaling in morphogen gradient read-out. *Molecular systems biology* **6** (2010).

146. Schwank, G. *et al.* Formation of the long range Dpp morphogen gradient. *PLoS biology* **9,** e1001111 (2011).

147. Aegerter-Wilmsen, T., Aegerter, C. & Bisseling, T. Model for the robust establishment of precise proportions in the early *Drosophila* embryo. *Journal of theoretical biology* **234,** 13–19 (2005).

148. Perkins, T., Jaeger, J., Reinitz, J. & Glass, L. Reverse engineering the gap gene network of Drosophila melanogaster. *PLOS computational biology* **2,** e51 (2006).

149. Neumann, C. J. & Cohen, S. M. Long-range action of Wingless organizes the dorsal-ventral axis of the Drosophila wing. *Development* **124,** 871–880 (1997).

150. Brower, D. L. & Jaffe, S. M. Requirement for integrins during Drosophila wing development. *Nature* **342,** 285–287 (1989).

151. Johnston, L. A. & Sanders, A. L. Wingless promotes cell survival but constrains growth during *Drosophila* wing development. *Nature cell biology* **5,** 827–833 (2003).

152. Meyer, F. Topographic distance and watershed lines. *Signal processing* **38,** 113–125 (1994).

153. Kanda, T., Sullivan, K. & Wahl, G. Histone–GFP fusion protein enables sensitive analysis of chromosome dynamics in living mammalian cells. *Current Biology* **8,** 377–385 (1998).

154. Campos-Ortega, J. A. & Hartenstein, V. *The embryonic development of Drosophila melanogaster* (Springer, 1997).

155. Czermin, B. *et al. Drosophila* Enhancer of Zeste/ESC Complexes Have a Histone H3 Methyltransferase Activity that Marks Chromosomal Polycomb Sites. *Cell* **111,** 185–196 (2002).

156. Al-Kofahi, Y., Lassoued, W., Lee, W. & Roysam, B. Improved automatic detection and segmentation of cell nuclei in histopathology images. *Biomedical Engineering, IEEE Transactions on* **57,** 841–852 (2010).

157. Wienert, S. *et al.* Detection and Segmentation of Cell Nuclei in Virtual Microscopy Images: A Minimum-Model Approach. *Scientific Reports* **2** (2012).

158. Bamford, P. & Lovell, B. Unsupervised cell nucleus segmentation with active contours. *Signal Processing* **71,** 203–213 (1998).

159. Goldberg, D. E. & Holland, J. H. Genetic algorithms and machine learning. *Machine Learning* **3,** 95–99 (1988).

160. Back, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms* (Oxford University Press, USA, 1996).

161. Kennedy, J. & Eberhart, R. in *Neural Networks, 1995. Proceedings., IEEE International Conference on* **4** (1995), 1942–1948.

162. Shi, Y. & Eberhart, R. in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on* (1998), 69–73.

163. Hansen, N., Müller, S. D. & Koumoutsakos, P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation* **11,** 1–18 (2003).

164. Hansen, N. The CMA evolution strategy: a comparing review. *Towards a new evolutionary computation,* 75–102 (2006).

165. Kennedy, J. in *Encyclopedia of Machine Learning* 760–766 (Springer, 2010).

166. Suzuki, T., Fujikura, K., Higashiyama, T. & Takata, K. DNA staining for fluorescence and laser confocal microscopy. *Journal of Histochemistry & Cytochemistry* **45,** 49–53 (1997).

167. Bansal, M., Belcastro, V., Ambesi-Impiombato, A. & Di Bernardo, D. How to infer gene networks from expression profiles. *Molecular systems biology* **3** (2007).

168. Äijö, T. & Lähdesmäki, H. Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. *Bioinformatics* **25,** 2937. ISSN: 1367-4803 (2009).

169. Yu, J., Smith, V., Wang, P., Hartemink, A. & Jarvis, E. Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*. ISSN: 1367-4803 (2004).

170. Enright, M. C., Day, N. P., Davies, C. E., Peacock, S. J. & Spratt, B. G. Multilocus sequence typing for characterization of methicillin-resistant and methicillin-susceptible clones ofStaphylococcus aureus. *Journal of clinical microbiology* **38,** 1008–1015 (2000).

171. Goentoro, L. A. *et al.* Quantifying the Gurken Morphogen Gradient in< i> Drosophila</i> Oogenesis. *Developmental cell* **11,** 263–272 (2006).

172. Kanodia, J. S. *et al.* Dynamics of the Dorsal morphogen gradient. *Proceedings of the National Academy of Sciences* **106,** 21707–21712 (2009).

173. Dewar, M. A., Kadirkamanathan, V., Opper, M. & Sanguinetti, G. Parameter estimation and inference for stochastic reaction-diffusion systems: application to morphogenesis in D. melanogaster. *BMC Systems Biology* **4,** 21 (2010).

174. Nellen, D., Burke, R., Struhl, G. & Basler, K. Direct and long-range action of a DPP morphogen gradient. *Cell* **85,** 357–368 (1996).

175. Gritli-Linde, A., Lewis, P., McMahon, A. P. & Linde, A. The whereabouts of a morphogen: direct evidence for short-and graded long-range activity of hedgehog signaling peptides. *Developmental biology* **236,** 364–386 (2001).

176. Lander, A. D., Nie, Q. & Wan, F. Y. Do morphogen gradients arise by diffusion? *Developmental cell* **2,** 785–796 (2002).

177. Bollenbach, T, Kruse, K, Pantazis, P, González-Gaitán, M & Jülicher, F. Robust formation of morphogen gradients. *Physical review letters* **94,** 018103 (2005).

178. Han, C., Yan, D., Belenkaya, T. Y. & Lin, X. Drosophila glypicans Dally and Dally-like shape the extracellular Wingless morphogen gradient in the wing disc. *Development* **132,** 667–679 (2005).

179. Hufnagel, L., Kreuger, J., Cohen, S. M. & Shraiman, B. I. On the role of glypicans in the process of morphogen gradient formation. *Developmental biology* **300,** 512–522 (2006).

180. He, F., Balling, R. & Zeng, A.-P. Reverse engineering and verification of gene networks: principles, assumptions, and limitations of present methods and future perspectives. *Journal of biotechnology* **144,** 190–203 (2009).

181. Hecker, M., Lambeck, S., Toepfer, S., Van Someren, E. & Guthke, R. Gene regulatory network inference: data integration in dynamic models—a review. *Biosystems* **96,** 86–103 (2009).

182. Bansal, M., Belcastro, V., Ambesi-Impiombato, A. & Di Bernardo, D. How to infer gene networks from expression profiles. *Molecular systems biology* **3** (2007).

183. Leicht, E. A. & Newman, M. E. Community structure in directed networks. *Physical review letters* **100,** 118703 (2008).

184. Polikar, R. Ensemble based systems in decision making. *Circuits and Systems Magazine, IEEE* **6,** 21–45 (2006).

185. Rokach, L. Ensemble-based classifiers. *Artificial Intelligence Review* **33,** 1–39 (2010).

186. Delgado-Gonzalo, R., Chenouard, N. & Unser, M. Spline-Based Deforming Ellipsoids for Interactive 3D Bioimage Segmentation (2013).

187. Keller, P. J. Imaging Morphogenesis: Technological Advances and Biological Insights. *Science* **340** (2013).

188. Huynh-Thu, V., Irrthum, A., Wehenkel, L., Geurts, P. & Isalan, M. Inferring Regulatory Networks from Expression Data Using Tree-Based Methods. *PLoS ONE* **5,** e12776 (2010).

189. Mises, R. & Pollaczek-Geiringer, H. Praktische Verfahren der Gleichungsauflösung. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik* **9,** 58–77 (1929).

190. Golub, G. H. & Van der Vorst, H. A. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics* **123,** 35–65 (2000).

191. Franceschet, M. PageRank: Standing on the shoulders of giants. *Communications of the ACM* **54,** 92–101 (2011).

192. Ipsen, I. & Wills, R. in *7th IMACS international symposium on iterative methods in scientific computing, Fields Institute, Toronto, Canada* **5** (2005).

193. Gubernatis, J. & Booth, T. Multiple extremal eigenpairs by the power method. *Journal of Computational Physics* **227,** 8508–8522 (2008).

194. Golub, G. H. & Van Loan, C. F. *Matrix computations* (JHUP, 2012).

195. LaMacchia, B. & Odlyzko, A. Solving large sparse linear systems over finite fields. *Advances in Cryptology-CRYPT0'90,* 109–133 (1991).

196. Golub, G. & Loan, C. V. *Matrix computations* chap. 9 (Johns Hopkins Univ Pr, 1996).

197. Kuczyński, J. & Woźniakowski, H. Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM journal on matrix analysis and applications* **13,** 1094 (1992).

198. Arora, S., Hazan, E. & Kale, S. in *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on* (2005), 339–348.

199. Tanimoto, H., Itoh, S., ten Dijke, P. & Tabata, T. Hedgehog Creates a Gradient of DPP Activity in *Drosophila* Wing Imaginal Discs. *Molecular cell* **5,** 59–71 (2000).

200. Persson, U. *et al.* The L45 loop in type I receptors for TGF-β family members is a critical determinant in specifying Smad isoform activation. *FEBS letters* **434,** 83–87 (1998).

201. Kühnlein, R. *et al.* spalt encodes an evolutionarily conserved zinc finger protein of novel structure which provides homeotic gene function in the head and tail region of the Drosophila embryo. *The EMBO journal* **13,** 168 (1994).

202. Shen, J., Dahmann, C. & Pflugfelder, G. Spatial discontinuity of Optomotor-blind expression in the Drosophila wing imaginal disc disrupts epithelial architecture and promotes cell sorting. *BMC developmental biology* **10,** 23 (2010).

203. Ninov, N. *et al.* Dpp signaling directs cell motility and invasiveness during epithelial morphogenesis. *Current biology* **20,** 513–520 (2010).

204. Kloeden, P., Platen, E. & Schurz, H. *Numerical solution of SDE through computer experiments* (Springer, 1994).

205. Lamba, H. Stepsize control for the Milstein scheme using first-exit-times.

206. Abe, K., Shaw, W. & Giles, M. Pricing Exotic Options using Local, Implied and Stochastic Volatility obtained from Market Data (2004).

207. Panzar, L. & Cipu, E. Using of stochastic Ito and Stratonovich integrals derived security pricing (2005).

208. Van Kampen, N. *Stochastic processes in physics and chemistry* (North-Holland, 2007).

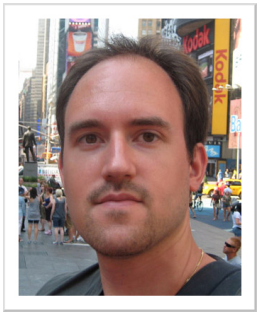
209. Higham, D. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM review,* 525–546 (2001).

210. Picchini, U. *SDE Toolbox: Simulation and Estimation of Stochastic Differential Equations with MATLAB* ().

211. Gilsing, H. & Shardlow, T. SDELab: A package for solving stochastic differential equations in MATLAB. *Journal of Computational and Applied Mathematics* **205,** 1002–1018 (2007).

212. Burrage, P. Runge-Kutta methods for stochastic differential equations (1999).

213. Honeycutt, R. Stochastic runge-kutta algorithms. I. White noise. *Physical Review A* **45,** 600–603 (1992).



Thomas Schaffter

Personal Data

Age 28
Nationality Swiss
Email thomas.schaffter@gmail.com
Website tschaffter.ch

Professional Experience

2008–Present **Ecole Polytechnique Fédérale de Lausanne** (full-time, 4 years 11 months)

Role: Researcher developing methods and computational tools for reverse engineering gene regulatory networks. This project is funded by the swiss initiative in systems biology (SystemsX.ch).
Achievements: Developed a novel and comprehensive method for *in silico* benchmark generation and performance profiling of network inference methods, and implemented it as an easy-to-use and open-source software called GeneNetWeaver. In colaboration with MIT, GNW was used to organize three editions of the DREAM competition, a community-wide network inference challenge.
Website: tschaffter.ch/projects/gnw

Role: Researcher developing algorithms for detecting clusters or modules in networks.
Achievements: Developed and profiled the performance of several modularity detection algorithms, and released them as an extensible and open-source Java toolkit. The algorithms developed have been applied to identify functional modules in the fruit fly (*Drosophila*) protein interaction map.
Website: tschaffter.ch/projects/jmod

Role: Researcher developing methods and computational tools for automatic detection and quantification of biological systems.
Achievements: Initiated and supervised a collaboration between EPFL and the University of Basel. Designed and implemented an open-source image processing toolbox for unsupervised detection and segmentation of the *Drosophila* wing and embryo, quantification of gene and protein expression, and cell nuclei detection in 3D confocal images. The data collected using this method was used to provide new insights into the development of the *Drosophila* wing.
Website: tschaffter.ch/projects/wingj

Role: Researcher developing a system for tracking heterogeneous groups of *Drosophila*.
Achievements: Developed and implemented a real-time C++/Qt application for helping biologists to track *Drosophila* and learn more about their behavior. The system requires two FireWire cameras and LEDs to track *Drosophila* and automatically infer their genetic identity.
Website: tschaffter.ch/projects/squid

Role: Teaching assistant in the master level course *Bio-Inspired Artificial Intelligence*.
Achievements: Prepared and introduced several labs on evolutionary algorithms, artificial neural networks, and evolutionary robotics. Supervised 10 semester and master student projects, each of them involving a part of research and the implementation of a software solution. Some projects were conducted in collaboration with other EPFL labs and universities.

Role: System administrator at the Laboratory of Intelligent Systems.
Achievements: Provided support for 40-50 collaborators and students, responsible for purchasing and maintaining IT equipment.

Education

2008–2013 **PhD in Biotechnology and Bioengineering**
Laboratory of Intelligent Systems (LIS)
Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland.
Specialization: Computational biology, artificial intelligence, optimization algorithms, graph and network theory, image analysis, information visualization and human-computer interaction.

2003–2008 **BSc and MSc in Microengineering**
Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland.
Specialization: Robotics and autonomous systems.

Awards & Features

Bioinformatics article "GeneNetWeaver: *In silico* benchmark generation and performance profiling of network inference methods" **has been ranked by *Faculty of 1000* among top 2% articles on biology and medicine**.

Received the **Best-poster award** at the 2008 exhibition of microengineer's master projects (>90 graduates).

Skills

Soft Skills Creative, organized, analytical, proactive.

IT **Software development** (Java, C, C++/Qt, Python), **web development** (HTML, PHP, MySQL, jQuery, WebKit), **IDEs** (Eclipse, Qt Creator, KDevelop), **drawing and illustrating** (Adobe Photoshop, Illustrator), **3D modeling and animation** (Blender, Pro/ENGINEER), **video editing** (Adobe Premiere, After Effects, VirtualDub), **data analysis and simulation** (Matlab, R, Matematica), **collaborative tools** (SVN, Git, Redmine), NX, **office suites** (Microsoft Office, LibreOffice), LaTeX, Bibtex, broad experience in Linux, Windows, and Mac OS, hardware skills.

Languages

French Native proficiency
English Full professional proficiency (C1)
German Elementary proficiency (A2)

Publications

Journal papers

T. Schaffter, R. Degado-Gonzalo, F. Hamaratoglu, M. Unser, M. Affolter, D. Floreano. Towards unsupervised and systematic quantification of biological systems, *In preparation*.

P. Ramdya, **T. Schaffter**, R. Benton, D. Floreano. Fluorescence Behavioral Imaging (FBI) tracks identity in heterogeneous groups of *Drosophila*, *PLOS ONE* 7(11), e48381, 2012.

T. Schaffter, D. Marbach, D. Floreano. GeneNetWeaver:*In silico* benchmark generation and performance profiling of network inference methods, *Bioinformatics* 27(16), 2263-2270, 2011. Featured by *Faculty of 1000*.

D. Marbach, R.J. Prill, **T. Schaffter**, C. Mattiussi, D. Floreano, G. Stolovitzky. Revealing strengths and weaknesses of methods for gene network inference, *Proceedings of the National Academy of Sciences USA* 107(14), 6286-6291, 2010.

D. Marbach, **T. Schaffter**, C. Mattiussi, D. Floreano. Generating realistic *in silico* gene networks for performance assessment of reverse engineering methods, *Journal of Computational Biology* 16(2), 229-239, 2009.

Theses

T. Schaffter. Reverse engineering of the *Drosophila* wing gene regulatory network, *PhD Thesis*, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2013 (expected).

T. Schaffter. Scalable Reverse Engineering of Nonlinear Gene Networks, *Master Thesis*, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2008. Received the *Best-poster award* at the exhibition of microengineer's master projects.

Other Activities

Sports	Badminton and snowshoeing.
Other	Japanese culture, 3D modeling and animation, robotics.
2005–2007	President of the Association Lausannoise des Universitaires Jurassiens (ALUJ).

Publications

Journal papers

T. Schaffter, R. Degado-Gonzalo, F. Hamaratoglu, M. Unser, M. Affolter, D. Floreano. Towards unsupervised and systematic segmentation of biological systems, *In preparation*.

P. Ramdya, **T. Schaffter**, R. Benton, D. Floreano. Fluorescence Behavioral Imaging (FBI) tracks identity in heterogeneous groups of *Drosophila*, *PLOS ONE* 7(11), e48381, 2012.

T. Schaffter, D. Marbach, D. Floreano. GeneNetWeaver:*in silico* benchmark generation and performance profiling of network inference methods, *Bioinformatics* 27(16), 2263-2270, 2011. Featured by *Faculty of 1000*.

D. Marbach, R.J. Prill, **T. Schaffter**, C. Mattiussi, D. Floreano, G. Stolovitzky. Revealing strengths and weaknesses of methods for gene network inference, *Proceedings of the National Academy of Sciences USA* 107(14), 6286-6291, 2010.

D. Marbach, **T. Schaffter**, C. Mattiussi, D. Floreano. Generating realistic *in silico* gene networks for performance assessment of reverse engineering methods, *Journal of Computational Biology* 16(2), 229-239, 2009.

Theses

T. Schaffter. Scalable Reverse Engineering of Nonlinear Gene Networks, *Master Thesis*, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2008. Received the *Best-poster award* at the exhibition of microengineer's master projects.