TOWARDS EVOLUTIONARY DESIGN OF INTELLIGENT TRANSPORTATION SYSTEMS

A. Martinoli[1], Y. Zhang[1,2], P. Prakash[1], E. K. Antonsson[2], and R. D. Olney[3]

[1] Collective Robotics Group, California Institute of Technology, Pasadena, CA 91125, U.S.A.
[2] Engineering Design Research Laboratory, California Institute of Technology, Pasadena, CA 91125, U.S.A.
[3] Delphi Delco Electronic Systems, Malibu, CA 90265, U.S.A.

Abstract. Intelligent transportation systems consisting of hundreds of individual vehicles which sense, decide, and act in the same-shared environment can be designed and controlled in two fundamentally different ways, using a centralized or distributed approach. The centralized approach implies an external system taking over the control of the vehicles and coordinating them, for instance by forming platoons, so that safety and fluidity are maintained. The distributed approach, instead, does not rely on any external control system and leaves the decisional autonomy to the individual vehicles. While the former approach has been shown to achieve a great degree of reliability, the latter represents an extremely appealing alternative because of its scalability and possible use in environments not endowed with external control systems. However, our human intuition and current engineering methods are not well adapted to design such distributed systems. In particular, when a certain group (or macroscopic) behavior is targeted, reverse engineering the individual (or microscopic) behavior is a non-trivial process. This process is even more complex and characterized by higher reliability requirements when each unit consists of a smart vehicle and a human being. This paper addresses design issues that could arise in the distributed approach and proposes an evolutionary computation methodology in order to help our intuition to devise the desired solutions. A simple case study concerning the placement and assessment of the number of proximity sensors is presented to support the discussion.

1 Introduction

Tremendous progress has been made in the last few decades in transportation systems safety. Particularly in automotive engineering, improvement in passive features such as seat belts, air bags, braking systems, traction control, crush zones, visibility, and lighting have dramatically reduced the accident fatality rate. For instance, in the U.S. the fatality rate per hundred million vehicle miles traveled has fallen from 5.5 to 1.5 in the period from the mid 1960's to the end of the decade. However, each year motor vehicle crashes in the U.S. still account for more than a staggering 40,000 deaths, 3,000,000 injuries, and $150 billion in financial losses [15]. More recently, the automotive industry has begun to incorporate several emerging new technologies such as radar techniques, vision, wireless communication, and robotic control in order to enhance automotive safety. Active safety systems offer the potential to achieve further dramatic reductions in accident fatalities, injuries, and property damage.

One of the active safety systems that has been proposed and intensively investigated in the past makes use of an external, centralized system to control platoons of vehicles on highways [14]. Although recent tests have proven the reliability of such systems, they are characterized by several intrinsic limitations. First, while this approach could represent a feasible solution for a highway, it is unlikely that the entire network will be endowed with enough external systems to be able to actively control each vehicle at each moment. Second, these active safety systems should not prevent those drivers who are controlling their vehicle safely from enjoying the pleasure of driving. Third, although similar liability problems sometimes arise in current chain accidents on highways, in the case of a mass failure of a centrally controlled road the primary responsibility would probably be attributed to the external safety system alone, with extremely high insurance costs for the company in charge.

An alternative solution for increasing traffic safety and fluidity without limiting the decisional autonomy of the individual driver, particularly during safe driving, is that based on a fully distributed approach. In this scenario, the active safety system is embedded uniquely in the vehicles and no external control is required. Of course, external traffic monitoring units, if available on a particular road, could communicate with single vehicles for suggesting itineraries or appropriate driving behavior under specific traffic conditions but the real

safety net would be represented exclusively by the vehicles themselves. These smart vehicles would therefore be endowed with technologies such as object detection, collision warning, and ultimately collision avoidance by accident prediction and autonomous vehicle control (brake, throttle, and steering).

Designing smart vehicles that can be part of such a distributed system is a non-trivial task, and in particular how a vehicle has to act autonomously in emergency situations. In other words, this means that in such situations, the intelligent vehicle has to outperform the human control in order to be effective. However, while artificial sensors and actuators can easily achieve performances far beyond human capabilities, manual control is still superior in closing the perception-to-action loop, and in particular in processing real-time salient information about the scene and judging the criticality of the situation. The challenge is even more difficult, both for a human being as well as for an intelligent car, because usually the road is shared among several vehicles, each autonomously acting and each of them having only a local (partial) perception of the whole scene (conversely to the centralized approach mentioned above). This autonomy of action is the heart of the whole engineering problem: any distributed active safety system should force vehicles to collaborate rather than compete in emergency situations but at the same time protect the life of individual drivers. In some traffic scenarios, the conflict of interest between individuals and group could be relevant and the secret for solving it in real time will probably lie in performing proactive prevention, using warning and conservative driver overriding, so that the situation will never become too critical because of the autonomy left to a single, maybe aggressive, driver.

Similar problems have been investigated in distributed robotic systems, an area of robotics that has been growing more and more in the past decade. We strongly believe that some of the knowledge cumulated in this field could be useful for solving problems in distributed traffic systems as well. Although the distributed robotics community has not yet agreed on sound formal methods for designing and controlling such systems (in particular when the system size exceeds the usual four or five units) a series of interesting results and several promising methods (most of the time with a strong heuristic component) have been reported in the literature [1,10].

Distributed traffic systems present similarities and differences with distributed robotic systems. On one hand, a distributed traffic system is similar to a distributed robotic system because both of them share physical embedment in the real world and distributed real-time control based on noisy sensors and actuators. On the other hand, a distributed traffic system is different from a distributed robotic system in several ways. First, in a distributed robotic system, the human being is not part of the individual unit: the robot is fully autonomous and does not need a driver for moving around. Safety issues are therefore only relevant if human beings share the same environment with robots characterized by relevant weight or volume, something that is fairly rare to be found in the current distributed robotics literature. However, safety in such systems is a fundamentally different problem. Second, while distributed robotic systems, especially those consisting of more than a few units, are usually homogeneous (i.e. consisting of similar individual units[1]), traffic systems are highly heterogeneous mainly due to both the human component (different driving styles) and the artificial component (e.g. trucks, cars, motorcycles; cars of different models; traditional cars, smart cars; and so on) characterizing their individuals. Third, man-made, pre-established environmental constraints (e.g. traffic rules, mono-directional flow in a lane, road curvature) and mean speed of the vehicles are completely different from one platform to the other. For instance, a classical obstacle avoidance maneuver in robots implies some sort of sharp turn and movement inversion, something that would be difficult to perform at high speed and in a constrained environment such as that of a highway.

In this paper, we would like to introduce an evolutionary computation methodology based on Genetic Algorithms (GAs) for designing and optimizing distributed embodied systems. This methodology has been proposed in the framework of distributed robotic systems [9] and shows several characteristics that make it interesting for the type of challenges in the distributed traffic systems we mentioned above. First, the methodology works off-line: solutions are first designed and tested in simulation, and when an appropriate level of performance is reached, downloaded on real platforms. This implies realistic simulators are available but offers the advantage of preventing the test of unsafe solutions directly on real hardware. Second, the methodology is platform-independent: it can be applied for instance to both robots and intelligent cars. Third, the methodology is system-oriented: given some system constraints (e.g. behavior-based model of the driver, morphology of the vehicle, sensor technology) it can help to engineer other hardware features (e.g. sensor placement, number, and characteristics) as well as software rules (e.g. warning rules, overriding control rules). Indeed, GAs are powerful parallel optimization algorithms which can search a large, maybe noisy, parameter or rule space without assuming its continuity, as do gradient-based methods. Fourth, although machine-learning methods in general are computationally expensive, they are automatic and the engineering effort involved in comparison to the hand-coded design is minimized to the mathematical formulation of the desired

[1] We are neglecting the small hardware differences due to manufacturing inaccuracies common to both types of systems.

performance and (currently) to the encoding of the real problem in the search space of the algorithm. The choice of the algorithmic parameters is also up to the engineer but usually GAs are not too sensitive to the initial settings. Heuristic criteria based on experience are usually used to perform these choices.

Latter in the paper, we will introduce evolutionary computation methods, explain how canonical GAs work, and outline some ad hoc changes we integrated in order to increase GAs' efficiency in engineering distributed systems. We will also explain which simulation tools we are using and specifically how we encoded a certain sensory solution on a simulated vehicle embedded in a collective traffic scenario. And lastly, will discuss the results of the proximity sensor case study, outline the current limitations of the proposed methodology, and conclude the paper with a series of challenges that have to be faced in the future in order to improve the methodology's efficiency in more complex problems.

2 Evolutionary Computation Methods

Since the mid 1960s, computer scientists have been studying stochastic search and optimization techniques based on the biological principles of evolution. These techniques include genetic algorithms, evolutionary strategies, genetic programming, and evolutionary programming. Mathematically speaking, these methods follow the same general approach: the parameter space is explored in parallel starting from a random set of solutions, with a progressive concentration on the most promising solutions around a local, possibly global, extremum (i.e. a minimum or a maximum).

2.1 General principles and definitions in GA

In GAs, each solution or *phenotype* is represented by the coding of its key parameters, producing a *genotype* or *chromosome*. Usually, in artificial evolution, the coding map is a one-to-one transformation: a phenotypic parameter is coded into a single genotypic segment, the *gene*, which in turn, once decoded, defines a single phenotypic parameter. The genotype space can be either represented by string of bits, as originally proposed in [4], or real numbers, as we have done in the case study presented in this paper.

The set of genotypes, which we will also call *genetic individuals*, being explored at any given time is known as the *population*. Iteratively (or in parallel if the system allows it) each genotype in the population is decoded and evaluated during a given time lapse, called *evaluation span*. The quantitative result of this evaluation is called the *fitness function*, as it measures each individual solution's efficacy.

Once the whole set of solutions is tested, several *genetic operators* can be applied to create the population for the next iteration, called also a *generation*. First, solutions are ranked according to their fitness and a subset is selected (*selection*). Then a new population is obtained by recombining the chromosomes (*crossover*) and random perturbation (*mutation*) of the selected individuals. As we will show below, problem-specific operators can also be added to the evolutionary process. All operators are usually associated with specific parameters (e.g. probability of mutation, of crossover, percentage of selected individuals, and so on) that in turn define the set of algorithmic parameters characterizing the GA. The choice of these parameters determines the balance between exploitation (by recombination) and exploration (by mutation) of the parallel search.

The *life span* of an individual is determined by the number of generations the individual survives. An evolutionary run is terminated after a given number of generations or when a solution with the desired precision is reached. Convergence can be tested by observing the standard deviation of the fitness and the mean distance from the genotypical center of mass of a given population.

2.2 Genetic operators

The three main genetic operators mentioned above work as follows. The selection operator is used to determine which individuals proceed from one generation to the next. The most common selection operators are the *roulette wheel*, *rank*, and *elitist* selection [13]. In the roulette wheel selection, the probability that an individual proceeds to the next generation is proportional to the fitness of the individual. A scaling factor is often associated with the roulette wheel selection to enhance the distinction between individuals of low fitness and individuals of high fitness. Rank selection, as the name suggests, ranks the solutions according to their fitness and then probabilistically selects individuals according to their rank rather than to their relative fitness in the population. Finally, the elitist selection deterministically selects a certain number of the fittest individuals of the population and generates copies of them to replenish the new population. The number of selected individuals can be a pre-established value, a quota of the population, or variable so that the population of the successive generation is maintained constant.

Crossover is a reproductive operator. Two parents, A ($A = A_1 \ldots A_N$) and B ($B = B_1 \ldots B_N$) are taken, a crossover point i ($1<i<N$) is selected, and two children C and D are produced such that $C = A_1 \ldots A_{i-1} B_i \ldots B_N$ and $D = B_1 \ldots B_{i-1} A_i \ldots A_N$. Such operators are based on the idea that two sub-optimal solutions may be

combined in an appropriate manner to produce an optimal result. Several crossover operators can be found in the literature: *one-point* crossover, *two-points* crossover, BLX-α (specific for real numbers encoding [3]).

The mutation operator is designed to induce small perturbations in the population. While the specific implementation details vary, the idea is to change the value of a randomly selected gene. When the genes are a binary representation of the phenotype, the mutation could be as simple as toggling the value of a bit on the gene. For real valued genotypes, the mutation is usually perturbing the gene's value by adding or subtracting a certain amount or replacing directly the gene. In both cases, the new gene values are chosen from a given random distribution (e.g. Gaussian, uniform white noise; subset of the parameter range, whole parameter range).

In certain experiments of the case study presented below, we allow the genotypes of the population to be of variable length. In order to more efficiently handle this specific chromosome feature we have introduced *one-point insertion* and *one-point deletion* operators. These operators randomly select a point on the genotype where a gene can be inserted or removed. Notice that introducing such operators is not a requirement, since in principle starting from an initial population of random variable chromosome length we can obtain new chromosome lengths via recombination, as shown in subsection 3.3.

2.3 Ad hoc GAs for distributed, noisy problems

Although GAs represent robust optimization and design tools, several additional problems are introduced by the collective, embodied, and noisy features characterizing distributed robotic or traffic systems. The main problem is related to the credit assignment (or fitness evaluation) and has its sources in the partial perception of the world at the individual level as well as in the corresponding inconsistency of the reward. Credit assignment problems manifest themselves in the optimization of both individual and group performances.

Fig. 1 summarizes different policies proposed in the distributed robotics literature for evolving (or learning) individual systems [9]. They differentiate on three basic aspects: individual or group credit assignment, public (sharing of parameters among different genetic individuals tested on different robots) or private (no sharing), and homogeneous (forced team homogeneity, one genetic individual is downloaded into all the teammates) or heterogeneous teams (different teammates may have different genotypes during the same evaluation span).

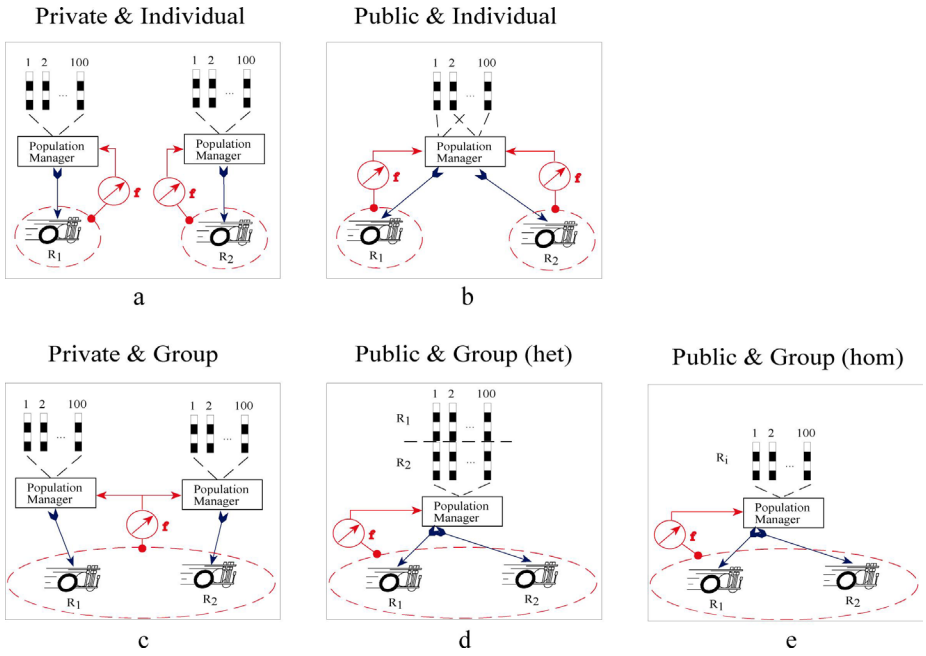

Figure 1: Five different policies to evolve system parameters distributed between two vehicles. The black and white strips represent the chromosomes and each population consists of 100 genetic individuals in this example.

In the optimization of the group performance, the credit assignment problem consists in the fact that if individuals have only partial perception of the whole collective system, individual fitness (Fig. 1a and 1b) may prompt for greedy solutions while group fitness may be difficult to be interpreted at the individual level (Fig. 1c and 1d). Furthermore, in an off-line optimization perspective, sharing genetic solutions among teammates is feasible through the population manager, but if combined with group evaluation and heterogeneous teams (see Fig. 1d), this approach is not scalable and becomes quickly computationally expensive if the algorithm has to explore a parameter space that is growing with the number of vehicles in the group. An alternative approach is that proposed in [5,16]: the credit assignment problem is bypassed by forcing team homogeneity and therefore the group score is evenly distributed among the teammates since all of them, being equal, have contributed on average equally to the team performance (see Fig. 1e). Unfortunately, this latter approach is not well adapted to distributed traffic systems because of the intrinsic heterogeneous nature of the system: we cannot force everyone to adopt the same driving style or ask a driver to control a truck as he would a sport car! The optimal

policy for this type of intrinsic heterogeneous systems is an open question in the distributed robotics area as well: the solution lies in finding the right balance between group and individual fitness and probably in forcing only some sort of categories rather than full homogeneity.

The problems are not simple either if we want to design an individual smart vehicle that can efficiently sense, decide, and act in a group. In other words, the key question is: how should we design a smart vehicle which has to share the road with a group of other vehicles which it cannot control, nor can it assume collaboration, nor can it necessarily exchange information with (e.g. standard cars)? Even if, as in the case study proposed here, there is no decision and/or action taken by the vehicle itself based on sensory input (and therefore no consequences of its individual decision on the group behavior), deciding how many sensors, which characteristics should they have, where should they be placed in order to achieve high monitoring performance and low cost, is not a trivial task, and in particular in a highly dynamic and noisy traffic scenario. Of course, if there exist no cost, no noise, no imprecision in manufacturing, and a limited set of free choices are associated with the sensors, a standard engineering hand-coded solution may suffice and outperform what an evolutionary algorithm may find.

As mentioned above, the main problem of evolving a vehicle embedded in a collective scenario is related to its partial perception and control of the whole system. This in turn results in solving the problem of how to represent the fitness noise generated by the rest of the system ("the others") at an individual level and how to eliminate as soon as possible from the population genetic solutions that received high fitness only because of lucky circumstances. One possible approach to this problem would be to enhance the selection reliability by increasing the number of evaluation spans per individual at each generation and then consider some central tendency value (e.g. mean, minimum) as a measure of their actual fitness. A fixed number of evaluations per individual would be easy to introduce in a standard GA loop but would add an additional algorithmic parameter (the number of evaluations per individual) and could be very computationally expensive, since evaluating solutions in this type of system usually requires much more time than applying genetic operators.

An alternative, more dynamic strategy would instead be to accumulate evaluations of the same individual over the whole evolutionary process and exploit them to derive a better judgment of its performance. The longer the life span of an individual, the higher the number of evaluations, and the more reliable is the judgment of its actual fitness. An additional strategy to fight against lucky individuals is to assume a conservative attitude. Thus, we would like to be stricter in accepting newborn individuals in the population and we would like to make sure that before they get accepted they are tested against their parents. Both of these strategies have prompted us to modify the canonical GA loop that forces a static size of the population (offspring of the most successful individuals simply replace parents) to a more dynamic population management during reproduction. The resulting modified GA loop is depicted in Fig. 2.

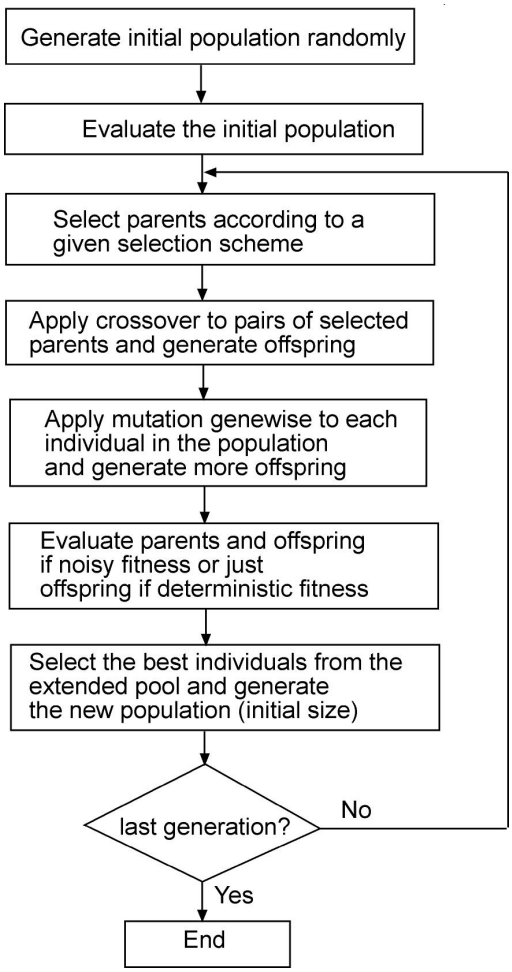

Figure 2: The modified GA loop used in the sensory configuration case study.

Following the flowchart depicted in Fig. 2 both parents and offspring are evaluated at each generation. The reproduction process is based on two different selection stages: the *parent* selection and the *new population* selection. The former selection choose parents for recombination while the latter selection is used to reduce the population size, extended during reproduction to a size that encompasses parents and children, to its default size before starting the new generation. The schemes for the two selections stages do not have to be the same. Children are evaluated only once at each generation while parents have been evaluated more than once from the generation they were created. The selection mechanism is therefore based on individuals that have been evaluated a different number of times. This dynamic evaluation approach is of course more computationally expensive than a canonical GA loop with selection and replacement of parents based on a single evaluation, but is more computationally efficient than systematically evaluating all offspring for a constant number of times at each generation, since multiple evaluations are reserved only for good individuals

which survive over generations. In order to assess the best individual at the last generation we however perform a fair test consisting of 100 evaluation spans on all distinct individuals belonging to the population.

3 A Case Study: Evolving Sensors' Number and Placement

As a first case study we have chosen the problem of automatically determining an optimal number of sensors and their placement on the vehicle in order to monitor a pre-established detection region. Although most of the details are oversimplified, a nice property of this problem is that in addition to representing a concrete task that engineers have to face in designing a smart vehicle, the evolved phenotypical geometry can be easily visualized on a simple 2D plot. This visualization is of great help for understanding why the evolutionary process has performed certain choices.

Fig. 3 illustrates the geometry of the problem. We consider round, unicycle (i.e. two motor wheels on an axis) vehicle. Sensors are characterized by a fixed range and cone of view (not shown in Fig. 3). Vehicle and sensors characteristics are faithful to robotic hardware available at Caltech [2]. The methodology is platform-independent and simulated vehicles could be easily transformed to emulate real cars endowed with automotive sensors.

3.1 Encoding

Sensors are mounted on the vehicle's body and their placement is characterized by two angles (see Fig. 3): φ (the angle between the front of the vehicle and the sensor's mount) and θ (the angle between the radius pointing to the sensor's mount and the sensor main axis). Each angle space is discretized in 400 intervals.

The number of sensors can be either pre-established (20) or variable (≥1). The chromosome length is therefore either constant (40) or variable (≥2). Experiments have been conducted forcing a left-right symmetry of the sensors' placement (in this case, the parameter space is reduced to half; sensors lying on the symmetry axis itself are mirrored on the opposite end) or by leaving their distribution unconstrained (asymmetric case). Object vehicles are considered to be detected if their centers are within the pre-established detection region (see subsection 3.3 for more details). An object vehicle is detected if one or more sensor rays penetrate the vehicle's body.

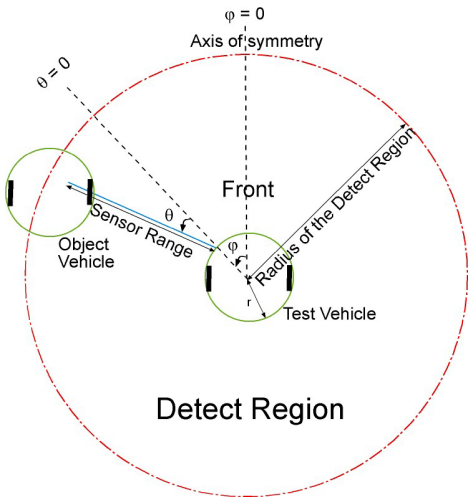

Figure 3: Parameters for sensor placement and detection region.

3.2 Evaluation tests

Four different types of evaluation are considered in this case study: a static, a quasi-static, a full coverage, and an embodied test. The reason for having multiple evaluation forms is two-fold. First, different forms of noise characterize the different tests and we are interested in understanding how noise shapes the evolved solutions. Second, each test has a different computational cost and we would like to minimize this cost or at least, as mentioned in subsection 2.3, find a way to quickly eliminate non-promising individuals using computationally cheap evaluations. For instance, on the same machine, in terms of effective real time required, the quasi-static evaluation span lasts five times longer than the static one, the full coverage test three times, and the embodied simulation 6685 times. Currently, we are evolving solutions using a single form of evaluation and we crosscheck the best individuals evolved according to one form using one of the other three tests. In the future, we may combine different tests in the evaluation span, following a hierarchical order in terms of the difficulty of the test. Fig. 4 and 5 show the details of the four tests.

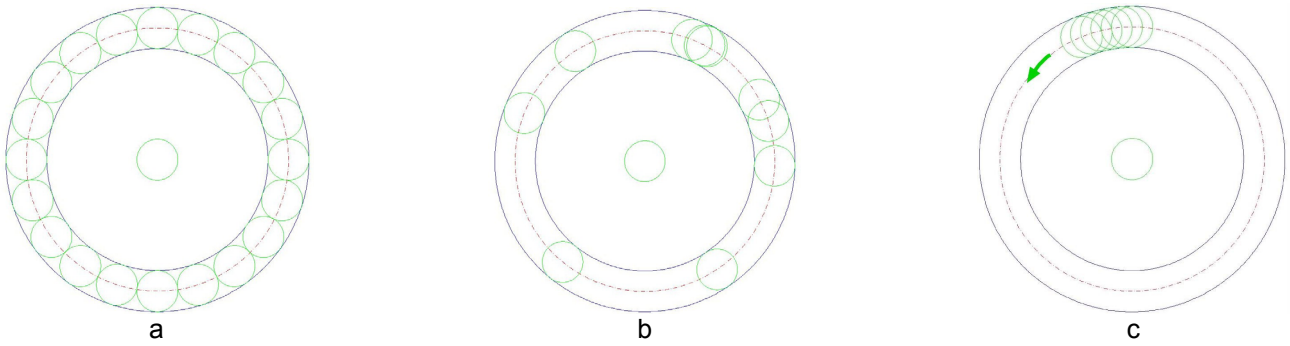

Figure 4: Graphical representation of the static test (a), the quasi-static test (b), and the full-coverage test (c). The test robot lies in the center and the object robots to be detected are distributed on a concentric, peripheral ring within the detection zone.

The static test is the most basic: a sensory solution is downloaded on the test vehicle while twenty object vehicles of the same dimensions are evenly distributed on an external ring (within the detection zone) without overlap (see Fig. 4a). Nothing is moving in this test: a single simulation step is therefore needed to assess how many of the object vehicles are detected with the current sensory configuration. In addition, since the evaluation environment is not changing over the whole evolutionary run, we do not need more than one single evaluation per individual and this represents its actual and deterministic fitness.

The quasi-static test is similar to the static test but the object vehicles are randomly positioned at each new evaluation span (see Fig. 4b). We spread 100 vehicles at a time on the peripheral ring, allowing overlap. Overlapped vehicles can be seen as a static version of moving vehicles for the coverage fitness we are interested in, in particular if we assume that, on average, having an evaluation span of 100 simulation steps with a single moving vehicle is similar to having all 100 vehicle's positions captured in one single time step. Conversely to the static test, for the same sensory configuration, the measured fitness may vary from evaluation to evaluation and different configurations may achieve the same fitness but based on two slightly different distributions of the object vehicles. Therefore, the longer the life span of the genetic individual over generations, the higher the number of cumulated evaluations spans, and the more faithful is the actual fitness. In this case study, we consider during both the evolutionary process and the final test (see subsection 2.3) the minimal value achieved during multiple evaluations as the statistical estimator for the actual fitness of a given individual. This choice is consistent with the safety task we are tackling: the sensory belt should insure a minimal amount of coverage and a given solution should therefore be judged based on the worst case.

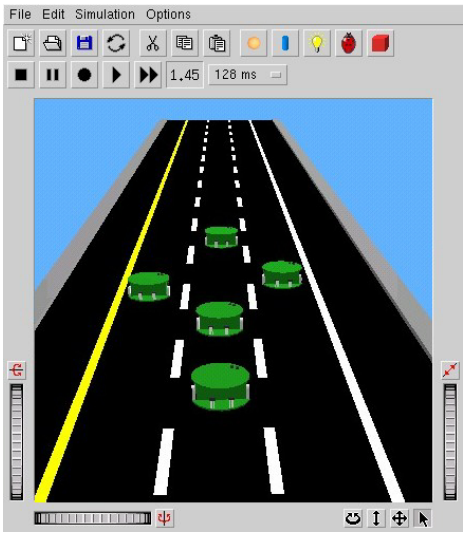

Figure 5: Screenshot of the embodied simulator.

The full coverage test is instead a systematic evaluation: a single vehicle is moving forward by one degree per simulation step along the periphery ring (see Fig. 4c). Therefore, the whole evaluation span lasts 360 simulation steps. Similar to the static case, one single evaluation of an individual represents its actual and deterministic fitness.

The embodied test is currently our final validation level and is performed using Webots [12], a sensor-based, realistic, 3D simulator (see Fig. 5). In the future real robots or real cars could represent the final test-bed for our evolved solutions. Sensors and actuators used in the embodied simulator are characterized by noise (± 10% white noise on the set wheel speed, ±1.5° random oscillation of the central beam axis of the proximity sensors[2]). Simple but realistic driver behaviors are implemented in the simulator for each vehicle. Drivers have different preferred cruise speeds, may or may not be allowed to change lanes, and may drive in high or low-density traffic. Their behavior is currently strictly reactive and at all times attentive: they keep the distance from the front vehicle, they keep the vehicle in the lane, and if they are allowed to pass another

[2] These values are based on the robotic platform available at Caltech [2] and have effectively delivered faithful results in other distributed robotic experiments.

vehicle they tend to do so only if their preferred speed is greater than the vehicle in front and no other vehicle is preventing the lane changing maneuver. Similar to the quasi-static test, the measured fitness may vary from evaluation to evaluation. Also in this case we use as statistical estimator of individual fitness the minimal value achieved over multiple evaluations, both during evolution and in the final test.

In order to more deeply investigate the differences in evolved sensor configurations due to environmental characteristics typical in distributed robotic experiments and in traffic scenarios, we have exploited two different Probability Density Functions (PDFs) which describe the probability of vehicle occurrences as a function of the angle of approach in both the quasi-static and full coverage tests. The first PDF represents a uniform distribution of the vehicle approaching angles in the detection zone, as would be the case in most of the distributed robotic experiments. Its graphical representation would assume a constant value between 0° and 360°. The second PDF represents a distribution of angle of approaches typical of a traffic scenario and can be generated from occurrence data recorded during a series of experiments using the embodied simulator (see Fig. 6 for two examples).

Similar to recent modeling methodologies we proposed in the area of distributed robotics [6,8,11], the key idea is to take advantage of more faithful (and therefore more computationally expensive) simulators such as Webots to more accurately model specific details at less computationally expensive levels of simulation. Although for this particular case study, we have taken into account only the direction of approach of other vehicles (mono-dimensional PDF), the same idea can be generalized to multiple dimensions (e.g. distance, orientation, relative speed) and other implementation levels (e.g. real vehicle data plugged into embodied simulations or in even more abstract levels of simulation). Finally, a given PDF has been used differently in the quasi-static and in the full coverage tests. While in the quasi-static test (a stochastic test) both PDFs have been used to bias distributions of vehicles on the periphery ring, in the full-coverage test (a deterministic test) the same PDFs have been used to weigh the fitness, as explained in the next subsection.

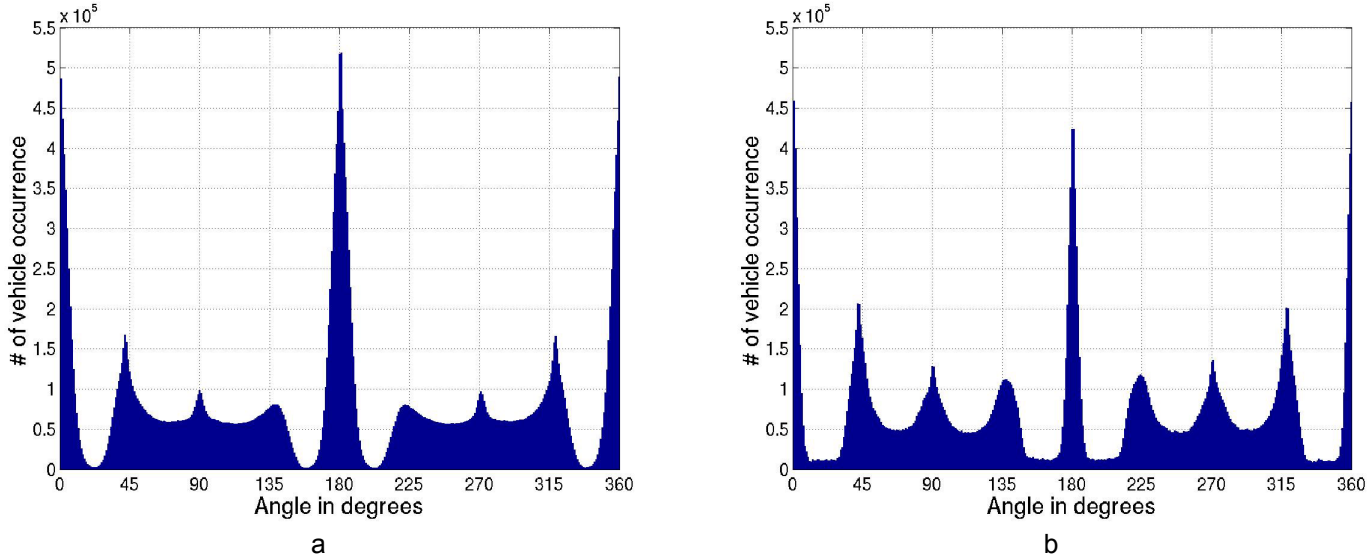

Figure 6: Vehicle occurrences as a function of the angle of approach of object vehicle. The data have been generated using the embodied simulator and represent the accumulative result of 5000 experiments, each of them characterized by the length of one evaluation span. The front of the vehicle is at 0°/360°. Drivers were characterized by different preferred cruiser speeds and were either not allowed to change lane (a) or allowed to do so (b). The corresponding PDFs can be generated from these data by normalizing the area under the curve to one.

3.3 GA and fitness function

The fitness function used is as follows:

$$f = \max\{(1-an)\sum_{i=1}^{V}\delta_i PDF(\alpha_i);0\} \qquad (1)$$

where *a* is the cost of a sensor, *n* is the number of sensors used in the current configuration, *V* is the number of vehicles effectively appeared in the detection region during the evaluation span, δ_i is either 1 or 0 as a function of the detection or not of the vehicle *i*. In the traffic scenario and under the full coverage test, the PDF used is derived directly from data generated with embodied simulations (see Fig. 6 for examples). In

uniform scenarios or under other test conditions, the PDF in the fitness function is independent of the angle and is therefore equal to *1/V* for all possible angles.

For the sensor configuration case study we used a parent selection based on roulette wheel, an elitist selection with variable number of selected individual (pruning the extended pool of solutions to the standard population size), one-point crossover, and a uniform mutation (i.e. random replacement of the gene with a value chosen from the whole parameter range). Insertion and deletion operators are additionally used in the experiments with variable number of sensors. Table 1 gives the numerical parameters used for the GA. The probability related to the genetic operators are meant to be static during an evolutionary run and calculated per genetic individual.

Population size	Selection scaling factor	$p_{crossover}$	$p_{mutation}$	$p_{insertion}$	$p_{deletion}$
50	2	0.2	0.182	0.05	0.05

Table 1: GA parameters

A special note concerning the crossover operator in experiments with variable chromosome length: if two chromosomes of different length recombine, we have to make sure that the crossover point is not biased and happening only between genes which are common to both chromosomes (i.e. crossover point limited by the length of the shorter chromosome). A possible solution is depicted in fig. 7.

Fig. 7 explains the effect of the one-point crossover operator at the phenotypical level. This could sound like an ad hoc solution for this particular sensor configuration problem but the same idea could be implemented at the genotypical level. Indeed, as shown in [7] variable length chromosomes can be represented as circular entities with a common alignment at the starting point. Therefore, this mechanism is general enough to be applied to other problems requiring variable length chromosomes. It is worth noticing that, as mentioned before, this type of crossover is able to generate new chromosome lengths without the intervention of insertion or deletion operators. For instance, in the example shown in Fig. 7, parents characterized by 11 and 5 sensors generate children with 7 and 9 sensors respectively. It is easy to verify the conservation of the total number of sensors after recombination.

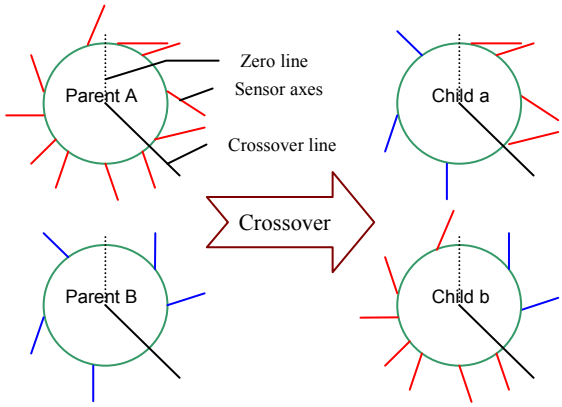

Figure 7: Illustration of the one-point crossover scheme for two chromosomes characterized by different lengths. Red and blue rays represent sensor axes on the test vehicle.

4 Results

Since we are currently still investigating the sensor placement case study, in this paper we report only a few samples of results that can be obtained with the methodology presented above. More extensive publication of results for this particular case study is foreseen in the near future.

Evolutionary runs based on the static, quasi-static, and full-coverage tests were repeated 20 times using different initializations of the random number generator and terminated after 200 generations. We ran each embodied evolutionary run only five times and stopped after 100 generations, a compromise due to the computational cost of this level of simulation. In histograms, unless otherwise stated, the height of a column represents the average performance while error bars correspond to the standard deviation over different runs of the same evolutionary experiment.

Fig. 8 shows an example of an evolutionary run performed using the full coverage test, a traffic PDF similar to that of Fig. 6a, and forcing symmetry on sensor placement. Fig. 8a shows the evolution of the population fitness over generations while Fig. 8b and c depict the best phenotypical solution at the last generation in the case of 20 sensors and with variable number of sensors respectively. It is interesting to notice how the variable number of sensors solution is able to cover about the same detection area with fewer sensors without having explicitly encoded in the fitness function any information of the type, size, speed of the other vehicles sharing the road.

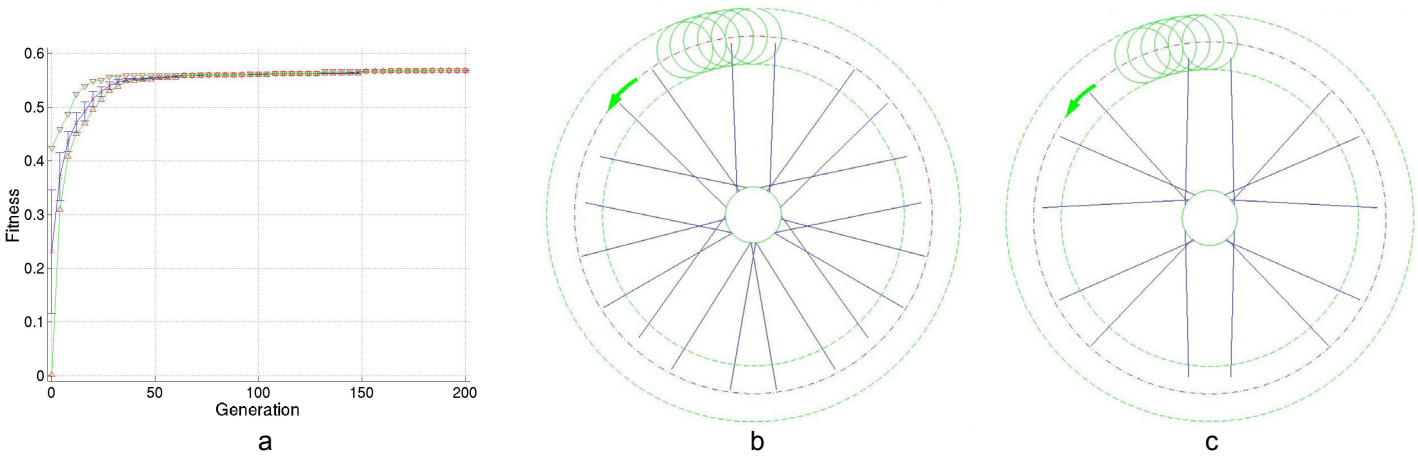

Figure 8: Example of data generated with a full-coverage evolutionary experiment under traffic conditions and enforced symmetry of the sensory configuration. a) Evolution of the population fitness over 200 generations (variable number of sensors). Error bars represent the standard deviation of the fitness among different individuals while lines with triangular markers represent the fitness of the best and worst individual at a certain generation. b) The phenotype of the best individuals after 200 generations are depicted in plot b (20 sensors) and c (variable number of sensors). The front of the vehicle is on the top of the picture.

Fig. 9 shows a comparison of the performance of the best individual (and often the most popular) at the last generation under different phenotypical constraints (symmetric or asymmetric configuration, 20 sensors or variable number of sensors) and under different final evaluations. Quasi-static and full-coverage evaluation used the non-lane changing PDF showed in Fig. 6a and traffic conditions in embodied simulations were the same as those used to record that PDF. Fig. 9a reports the results of a "native" final test for each evaluation form: for instance, if an individual was evolved in a quasi-static environment, the performance plotted was recorded during a final quasi-static test. Fig. 9b and c crosscheck the performance of individuals evolved in different environments under the same final evaluation. In Fig. 9b all the best individuals have been tested using the full coverage test while in Fig. 9c all the best individuals have been evaluated using an embodied final test. While quasi-static and full coverage tests are almost interchangeable, individuals evolved under embodied conditions do not perform well under a full coverage test. The opposite is also true but the drop in performance is less drastic: individual evolved under full-coverage or quasi-static conditions perform only slightly worse than individuals evolved in embodied simulation outlining the fact that a PDF based on a 5000 times longer evaluation span is a good estimator of the embodied conditions. The static simulation is clearly the simplest one and its best individuals are subject to relevant fitness drops when tested under other conditions.

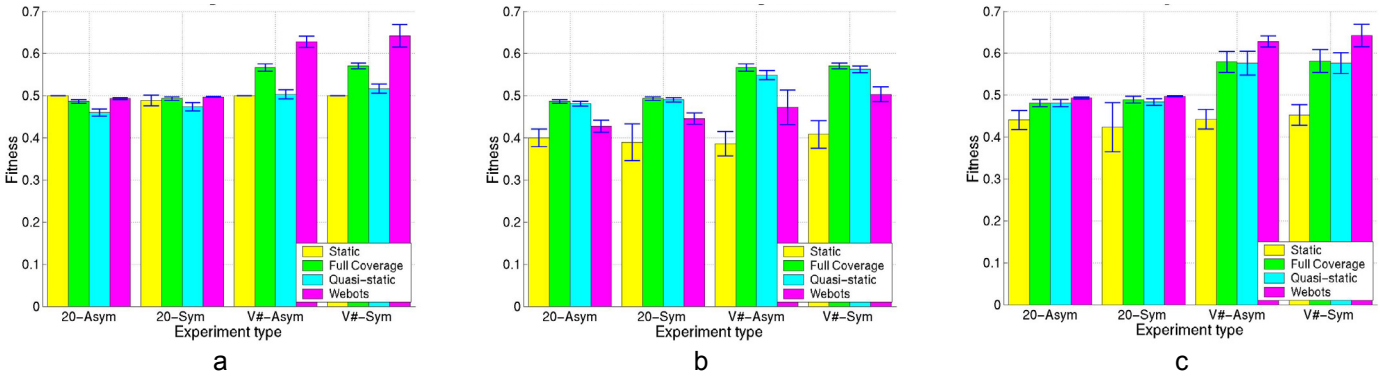

Figure 9: Performance of the best individuals of the last generations under the final test. Individuals evolved according to one specific evaluation form are re-evaluated under the same form in the final test (a). Individuals evolved according each of the four evaluation forms are then finally evaluated based on the full coverage test (b) or embodied test (c). As reference, it is worth noticing that, performances of individuals under the full coverage test in (a) and (b) are the same and similarly those of individuals under the embodied test in (a) and (c).

Furthermore, Fig. 9 shows also that enforcing symmetry does not necessarily improve the quality of performances achieved at the last generation. Enforcing symmetry (and therefore reducing the search space

to half) usually only improves convergence time but we do not see any major difference in performance at the end of evolutionary run since 200 generations is a long enough period for discovering good solutions in asymmetric cases.

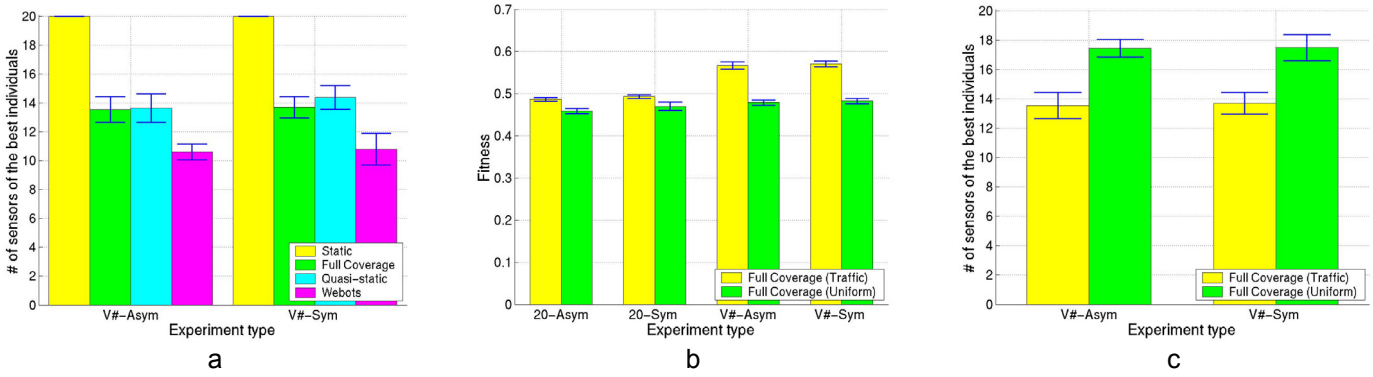

Figure 10: a) Number of sensors of the best individuals at the last generation of the experiments reported in Fig. 9. b) Comparison of the fitness of the best individuals in a full coverage final test with two different PDF (traffic and uniform distribution). PDF for a given evolutionary run are the same during evolution and in the final test. c) Number of sensors of the best individuals at the last generation in both experiments.

Instead, leaving the number of sensors to be used unconstrained allow individuals to improve their fitness in all cases but the static test. Fig. 10a shows the number of sensors used by the best individuals at the last generation and provides for us an explanation of the improved performances of variable chromosome-length individuals. While in the static experiments the 20 non-overlapping object vehicles can only be detected with 20 different sensors, the quasi-static, full-coverage, and embodied experiments push evolution to find more efficient solutions in traffic scenarios: individuals increase their fitness by reducing the number of sensors and therefore trading off a slightly reduced coverage with a more important reduction in the cost of the sensor configuration. It is worth noticing that the final number of sensors is undoubtedly conditioned by the arbitrary factor *a* (Eq. 1) which determines the balance between coverage efficiency and sensor cost.

Fig. 10b and c show the influence of a chosen PDF on the fitness obtained and the number of sensors used. The GA is smart enough to understand that fewer sensors are required in the traffic scenarios to cover the detection region as efficiently as it could with more sensors in an environment characterized by more uniform distribution of the object vehicles. In this case as well, symmetry forcing does not necessarily bring any advantage on the quality of the final solution but rather faster convergence due to reduced search space.

5 Conclusion and Outlook

In this paper we have presented an off-line, evolutionary computation methodology for designing distributed systems characterized by embodiment, noisy sensors and actuators, and fully distributed control. After having discussed which main challenges any machine-learning methodology has to face for automatic design of this type of system, we have described how a canonical GA can be modified in order to improve its efficiency in facing such challenges. Furthermore, as part of this methodology, we have proposed possible levels of simulation characterized by different abstraction and computational cost and shown how it is possible to move from one to the other. Finally, we have presented sample results obtained from a case study concerned with the design of the sensory configuration (number of sensors and placement) for monitoring purposes.

Although more work needs to be done in order to improve the computational efficiency of the methodology and to understand the role of noise and simulation level on the evolved solutions, we believe that the results reported in this paper are promising, in particular because we do not see any fundamental barrier in applying an incremental evolutionary approach to more complex problems. In the near future, we plan to introduce more realistic elements at the sensory and vehicle level and show that when the number of design parameters is large and when noise is involved, machine-learning solutions start to be competitive with traditionally engineered solutions from quality as well as from an engineering effort point of view.

Acknowledgements

This work was mainly supported by Delphi Delco Electronic Systems. Further funding was received from the Caltech Center for Neuromorphic Systems Engineering as part of the NSF Engineering Research Center program under grant EEC-9402726.

References

1. Bonabeau E., Dorigo M., and Theraulaz G., “Swarm Intelligence: From Natural to Artificial Systems”, SFI Studies in the Science of Complexity, Oxford University Press, New York, NY, 1999.
2. CORO: http://www.coro.caltech.edu/robots_and_tools.htm.
3. Eshelman L. J. and Schaffer J. D., “Real-coded genetic algorithms and interval-schemata”. *Foundations of Genetic Algorithms 2*, Morgan Kaufman Publishers, San Mateo, 1993, pp. 187-202.
4. Goldberg D. E., “Genetic Algorithms in Search, Optimization, and Machine Learning”. Addison-Wesley, Reading, MA, 1989.
5. Hayes A. T., Martinoli A., and Goodman R. M. “Swarm Robotic Odor Localization: Off-Line Optimization and Validation with Real Robots. Special Issue on Bio-Inspired Robotics, *Robotica*. To appear.
6. Ijspeert A. J., Martinoli A., Billard A., and Gambardella L.M., “Collaboration through the Exploitation of Local Interactions in Autonomous Collective Robotics: The Stick Pulling Experiment”. *Autonomous Robots*, Vol. 11, No. 2, pp. 149-171, 2001.
7. Lee C.-Y. and Antonsson E., “Variable Length Genomes for Evolutionary Algorithms”. *Proc. of the Genetic and Evolutionary Computation Conference*, Las Vegas, NV, 2000, p. 806.
8. Lerman K., Galstyan A., Martinoli A., Ijspeert A. J., “A Macroscopic Analytical Model of Collaboration in Distributed Robotic Systems”. *Artificial Life*, Vol. 7, No. 4, pp. 375-393, 2001.
9. Martinoli A., “Swarm Intelligence in Autonomous Collective Robotics: From Tools to the Analysis and Synthesis of Distributed Collective Strategies”. Unpublished doctoral manuscript, EPFL Ph.D. Thesis Nr. 2069, October, 1999, Lausanne, Switzerland, downloadable at http://www.coro.caltech.edu/people/alcherio/am_pub.html.
10. Martinoli A., “Collective Complexity out of Individual Simplicity”. Invited book review on "Swarm Intelligence: From Natural to Artificial Systems", by E. Bonabeau, M. Dorigo, and G. Theraulaz. *Artificial Life*, Vol. 7, No. 3, pp. 315-319, 2001.
11. Martinoli A. and Easton K., “Modeling Swarm Robotic Systems”. *Proc. of the Eight Int. Symp. on Experimental Robotics ISER-02,* Sant'Angelo d'Ischia*,* Italy, July, 2002. Springer Tracts in Advanced Robotics (2003), pp. 285-294.
12. Michel O., “Webots: Symbiosis Between Virtual and Real Mobile Robots”. In Heuding J.-C., editor, *Proc. of the First Int. Conf. on Virtual Worlds*, Paris, France, July, 1998, Springer Verlag, pp. 254-263. See also http://www.cyberbotics.com/webots/.
13. Mitchell M.,"An Introduction to Genetic Algorithms". The MIT Press, Cambridge, MA, 1996.
14. Partners for Advanced Transit and Highways (PATH): http://www.path.berkeley.edu/PATH/Publications/Default.htm.
15. US DOT NHTSA (United States Department of Transportation - National Highway Traffic Safety Administration), “Traffic Safety Facts 2000”, 2000.
16. Versino C. and Gambardella L. M.,“Learning Real Team Solutions”. In Weiss G., editor, *DAI Meets Machine Learning*, Lecture Notes in Artificial Intelligence, Springer Verlag, Berlin, 1997, pp. 40-61.