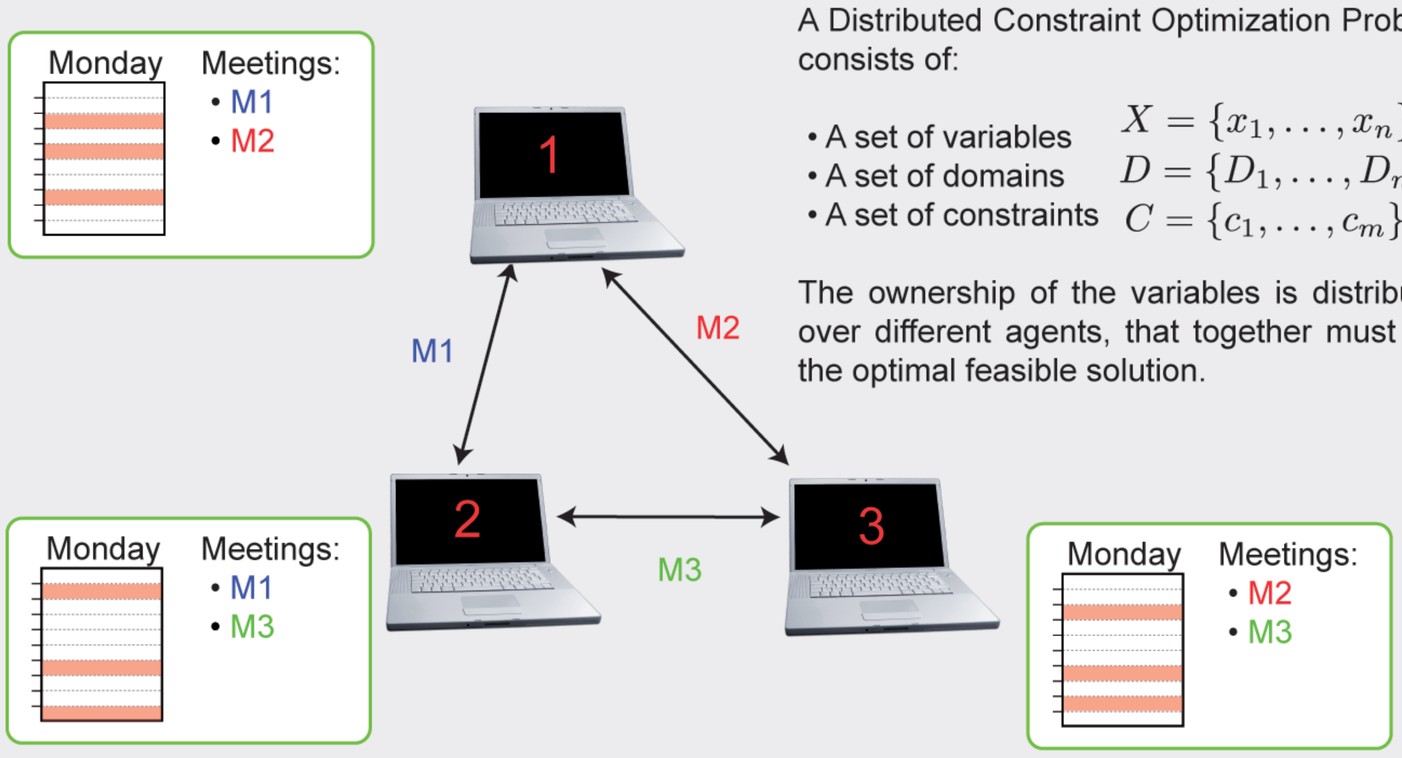
FRODO: An Open-Source Framework for Distributed Constraint Optimization

Distributed Constraint Optimization (DCOP)



A Distributed Constraint Optimization Problem consists of:

- A set of variables $X = \{x_1, \ldots, x_n\}$
- A set of domains $D = \{D_1, \ldots, D_n\}$
- A set of constraints $C = \{c_1, \ldots, c_m\}$

The ownership of the variables is distributed over different agents, that together must find the optimal feasible solution.

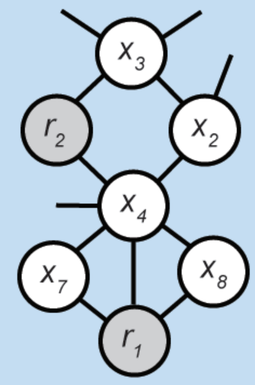
Handling Uncertainty

Stochastic DCOP

- Uncertainty is modeled by **independent random variables**
- **Exogenous uncertainty**: probability distributions do not depend on decision variables
- The goal is to **maximize the expected total utility**

E[DPOP] Algorithm

- **Sample** probability distributions to get an **optimality/complexity trade-off**
- Agents reasoning over inconsistent samples would yield low-quality solutions
- **Collaborative sampling**: all decision variables neighboring a given random variable collaborate to choose the same samples
- The **local influences of random variables** allow to distribute the sampling load by assigning it to the lowest common ancestor



Agent Autonomy

Many **DCOP** problems involve different, autonomous entities. In general such entities will be faced with time constraints or have to coordinate with agents that are slower. Reasonable requirements on a DCOP algorithm are therefore that:

- Every agent retains decision autonomy
- Every agent retains the ability to make a decision at any time during the runtime of the algorithm

ASODPOP is an asynchronous algorithm that provides these requirements by providing:

- **Fast information propagation** through the network
- **Speculative computation** to help agents make decisions based on partial information

Information Propagation

In order for an agent to cope with both time constraints and slower neighbors, the speed at which information is propagated in the network becomes important. In ASODPOP,

- Information is sent from **bottom to top**, in a **best-first order**
- Agents request information from their children by sending an **ASK message**
- Agents can respond with **partial information**, facilitating **fast information propagation**

Speculative Computation

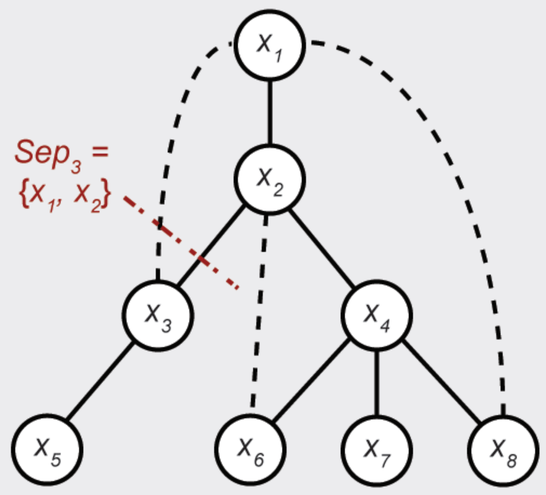
Due to the use of partial information, agents might not have all the information needed available to them when a decision must be made. When a group of agents solves the same type of problem multiple times, **speculative computation** can be used to assist the agents in making a decision.

Speculative computation can be used to both:

- guide agents to the optimal solution
- help agents avoid infeasible solutions.

Variable Hierarchy

Decision variables are ordered along a Depth-First Search pseudo-tree (**DFS**) in which edges are constraints.



Reacting to Changes

Dynamic DCOP

- Changes can occur in the problem:
  - In the constraints' utilities
  - In the topology of the graph
- The goal is to **repair the solution** so as to maximize the total utility, minus the **cost of repair**

RS-DPOP Algorithm

- Re-solve, **reusing previous computation results** to avoid re-sending identical messages
- Model the cost of repair
- Introduce **commitment deadlines** for decision variables

Protecting Privacy

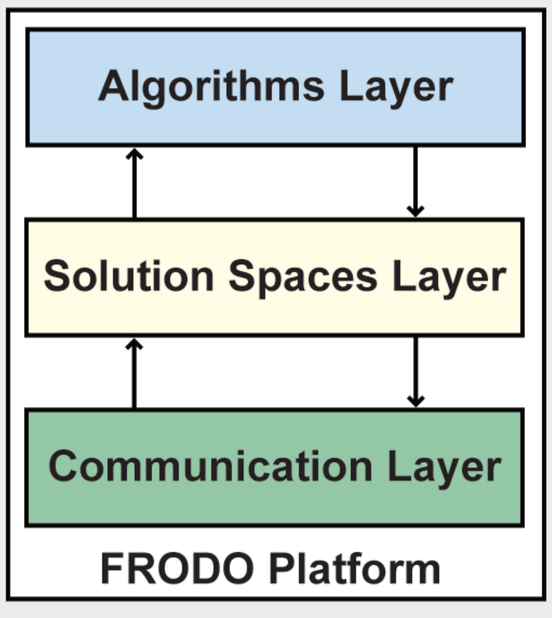
Potential Privacy Leaks

- ***Agent privacy***: agents' identities
- ***Topology privacy***: who has constraints with whom
- ***Constraint privacy***: preferences
- ***Decision privacy***: the decisions in the optimal solution found

Encrypted Computation

- **Codenames** protect the agents' identities and constraint scopes
- **Adding random numbers**, resp. **homomorphic encryption** yields partial, resp. full constraint privacy
- **Tree re-rooting** hides decisions

Software Architecture



- **Modular** implementations of various **DCOP algorithms**: DPOP, ADOPT, ASODPOP, E[DPOP], S-DPOP...
- Data structures for **constraints**
- Efficient operations on constraints: join, projection, composition...
- **Fully distributed** communication
- Support for TCP/IP
- **Extensive unit tests**
- Systematic code documentation

The FRODO Java source code is available under the **GNU Affero GPL license**:
http://liawww.epfl.ch/frodo/

JaCoP: Java Constraint Programming Solver

JaCoP is an **open-source** Java-based Constraint Programming solver that is available under the GNU Affero GPL license. It focuses on **modeling power**, **ease of extension**, **ease of maintenance**, and last but not least **efficiency**. Its major strength is the number of available **global constraints** with their specialized pruning/reasoning algorithms. It provides a flexible mechanism to specify search making it relatively easy to encode problem specific search procedures.

JaCoP allows the DCOP algorithms in FRODO to address problems in which agents have to **solve complex local subproblems**. In particular, it enables efficient generation and representation of Solution Spaces.

JaCoP is available at: http://jacop.eu/

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Artificial Intelligence Laboratory
Thomas Léauté, Brammert Ottens, Radoslaw Szymanek and Boi Faltings
{firstname.lastname}@epfl.ch
http://liawww.epfl.ch/frodo/