On the Deployment of Factor Graph Elements to Operate Max-Sum in Dynamic Ambient Environments

Pierre Rust[1,2] Gauthier Picard[1] Fano Ramparany[2]

[1]MINES Saint-Étienne, CNRS
Lab Hubert Curien UMR 5516

[2]Orange Labs

Overview

- Smart Environment Configuration Problem
- Deployment Problem for DCOP and SECP
- Dynamics in the Deployment Problem
- Experiments
- Future work

Smart Environment Configuration Problem

Decentralized coordination for smart homes

- Coordination among connected devices in the smart home : no central coordinator
- Fulfill user-defined rules and minimize energy consumption
- All computations are distributed directly on the connected devices: light bulbs, roller shutter, etc.
- Constrained devices
  - limited cpu and memory resources
  - limited communication capabilities

SECP Model

Actuators:
Connected light bulbs, TV, Rolling shutters, ...

Sensors:
Presence detector, Luminosity Sensor, etc.

Physical dependecy Models:
E.g. Living-room light model

User Preferences:
expressed as rules ;

```
IF    presence_living_room      =  1
AND   light_sensor_living_room  <  60
THEN  light_level_living_room   ←  60
AND   shutter_living_room       ←  0
```

SECP Model

Actuators:

- *Decision* Variable x_i, Domain $\mathbf{x}_i \in \mathcal{D}_{x_i}$
- Cost function$c_i : \mathcal{D}_{x_i} \rightarrow \mathbb{R}$

Sensors:

- *Read-only* Variable s_l, Domain $\mathbf{s}_i \in \mathcal{D}_{s_l}$

Physical dependecy Models:

- Give the expected state of the environment from a set of actuator-variables influencing this model
- Variable y_j representing the *expected* state of the environment
- Function $\phi_j : \prod_{\varsigma \in \sigma(\phi_j)} \mathcal{D}_\varsigma \rightarrow \mathcal{D}_{y_j}$

User Preferences:

- Utility fonction u_k
- Distance from the current expected state to the target state of the environnement

Formulating the SECP as a DCOP

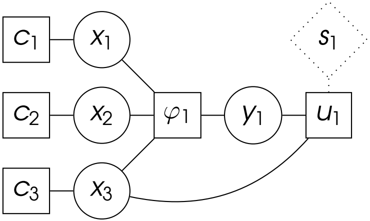
- Optimization problem

$$\underset{x_i \in \nu(\mathfrak{A})}{\textbf{minimize}} \sum_{i \in \mathfrak{A}} c_i \quad \text{and} \quad \underset{\substack{x_i \in \nu(\mathfrak{A}) \\ y_j \in \nu(\Phi)}}{\textbf{maximize}} \sum_{k \in \mathfrak{R}} u_k$$

$$\textbf{subject to} \quad \phi_j(x_j^1, \ldots, x_j^{\overline{\phi_j}}) = y_j \quad \forall y_j \in \nu(\Phi)$$

- Mono objective DCOP :

$$\underset{\substack{x_i \in \nu(\mathfrak{A}) \\ y_j \in \nu(\Phi)}}{\textbf{maximize}} \quad \omega_u \sum_{k \in \mathfrak{R}} u_k - \omega_c \sum_{i \in \mathfrak{A}} c_i + \sum_{\varphi_j \in \breve{}} \varphi_j$$



DCOP

Distributed Constraints Optimization Problem

A DCOP is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mu \rangle$, where:

- $\mathcal{A} = \{a_1, \dots, a_{|A|}\}$ is a set of agents;
- $\mathcal{X} = \{x_1, \dots, x_n\}$ are variables;
- $\mathcal{D} = \{\mathcal{D}_{x_1}, \dots, \mathcal{D}_{x_n}\}$ is a set of finite domains, for the x_i variables;
- $\mathcal{C} = \{c_1, \dots, c_m\}$ is a set of soft constraints, where each c_i defines a cost $\in \mathbb{R} \cup \{\infty\}$ for each combination of assignments to a subset of variables;
- μ is a function mapping variables to their associated agent.

A *solution* to the DCOP is an assignment to all variables that minimizes $\sum_i c_i$.

The mapping function

$\mu : \mathcal{X} \rightarrow \mathcal{A}$

- surjective function, from variable to agents
- assigns the control of each variable x_i to an agent $\mu(x_i)$

Common assumptions:

- each agent controls exactly one variable (bijection)
- binary constraints

Real distributed problems:

- agents must be hosted on real devices
- the set of devices might be given by the problem
- for some variables the relation with an agent is obvious, but not always

Real problems

Modelling real distributed problems

One agent for each variable:

- several agents on a single device
- how to decide on which device each agent should be hosted ?

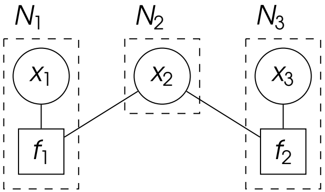
One agent for each device:

- one agent controls several variables
- how to decide which agent is responsible for each variable ?

Factor Graph algorithms

Factors also need to be deployed

- one computation for each variable
- one computation for each constraint (aka factor)



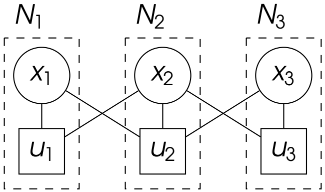
How to decide which agents should host the factors computations ?

Utility based Factor Graph

Max-Sum

Two possible factor graph modeling approaches:

- interaction-based factor graph
- utility-based factor graph



- Difficult for some problems
- Less efficient: add cycles, more factors, etc.
- Still does not solve the problem of abstract modeling variables !

Factor Graph Deployment Problem

For Smart Environment Configuration

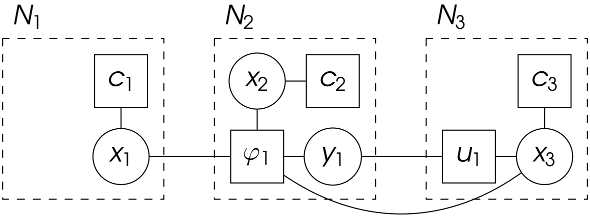
The Deployment problem:

- defining the mapping function μ.
- definition of *optimal* deployment: problem-dependent
- optimal deployment $\equiv$ graph partitioning : NP-hard !

Mathematical optimization problem : Integer Linear Program (for graph partitioning).

Deploying the SECP factor graph:

- Devices have limited memory
- Communication is expensive and has limited bandwidth
- Variable related to an actuator are hosted by it
- Objective : **minimize overall communication between agents**



Binary ILP for computation distribution

- **com**(x_i, f_j) : communication load between variable x_i and factor f_j
- **mem**(e) : memory footprint for a computation and **cap**(a_k) memory capacity for a device
- x_i^k ad f_i^k : binary variables that map factor graph elements to agents and for linearization purpose $\alpha_{ijk} = x_i^k \cdot f_j^k$
- fix actuactors variables and cost factors to be hosted by their owner
- extra constraints for memory capacity

Binary ILP for computation deployment

Constraints for Factor graph computations deployment

$$\underset{x_i^k, f_j^k}{\textbf{minimize}} \quad \sum_{(x_i,f_j)\in E} \sum_{a_k \in \mathcal{A}} \textbf{com}(x_i, f_j) \cdot (1 - \alpha_{ijk}) \tag{1}$$

subject to

$$\forall x_i \in V_x, \quad \sum_{a_k \in \mathcal{A}} x_i^k = 1 \tag{2}$$

$$\forall f_j \in V_f, \quad \sum_{a_k \in \mathcal{A}} f_j^k = 1 \tag{3}$$

$$\forall a_k \in \mathcal{A}, \quad \sum_{x_i \in V_x} x_i^k + \sum_{f_j \in V_f} f_j^k \geq 1 \tag{4}$$

$$\forall (x_i, f_j) \in E, \quad \alpha_{ijk} \leq x_i^k \tag{5}$$

$$\forall (x_i, f_j) \in E, \quad \alpha_{ijk} \leq f_j^k \tag{6}$$

$$\forall (x_i, f_j) \in E, \quad \alpha_{ijk} \geq x_i^k + f_j^k - 1 \tag{7}$$

Binary ILP for computation deployment

Constraints from SECP properties

$$\forall a_k \in \mathcal{A}, \forall x_i \in \rho_x^{-1}(a_k), \quad x_i^k = 1 \tag{8}$$

$$\forall a_k \in \mathcal{A}, \forall f_j \in \rho_f^{-1}(a_k), \quad f_j^k = 1 \tag{9}$$

$$\forall a_k \in \mathcal{A}, \quad \sum_{x_i \in V_x} \mathbf{mem}(x_i) \cdot x_i^k + \sum_{f_j \in V_f} \mathbf{mem}(f_j) \cdot f_j^k \leq \mathbf{cap}(a_k) \tag{10}$$

Solving the ILP for computation deployment

NP-hard, but can be solved with branch-and-cut.

But it's not distributed !

- It could be: distributed simplex
- Still probably too hard for our devices
- In SECP, computing power is available when bootstrapping the system
- Gives us a reference for optimality: benchmarking

SECP is a dynamic problem

Dynamics in the infrastructure:

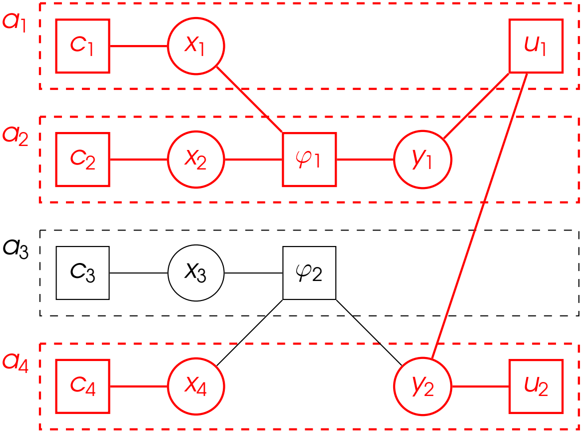
- Devices can disappear
- New devices can be added to the system

At run time:

- No powerful device available to solve the ILP
- The deployment must be repaired: self adaptation
- Only consider a portion of the factor graph: the neighborhood.

Notion of neighborhood

- The *neighborhood* of an agent a_k is the set of agents which hosts a computation linked to a computation hosted by a_k.
- The set of edges connected to the neighborhood : $E[a_k]$
- the set of neighborhood variables (resp. factors) : $V_x[a_k]$ (and $V_f[a_k]$)



Neighborhood $A[a_2] = a_1, a_2$ and a_4
Associated sets $E[a_2]$, $V_x[a_2]$ and $V_f[a_2]$

Adaptation to device arrival - ILP version

- Reuse the ILP for computation distribution
  - But restrict it to the neighborhood ot the new device.
  - Probably not optimal, but only requires local and limited knowledge of the SECP.
- Solving the reduced ILP:
  - Smaller problem : could be distributed on the agents from the neighborhood
  - Worst case: the new agents is connected to all other agents.

Adaptation to device arrival - Newcomer centric

Newcomer decision problem

- Newcomer centric approach:
  - the newcomer calls for proposals to move some computations
  - the newcomer choose a set of computations, based on their costs and its own memory capacity
- Each neighbor $a_\ell \in \mathcal{A}[a_k]$ sends its proposal $\langle V^{\ell \rightarrow k}, E^{\ell \rightarrow k}, \mathbf{com} \rangle$,
  - $V^{\ell \rightarrow k}$: proposed computations
  - $E^{\ell \rightarrow k}$ the edges connected to these computations
  - **com** the communication cost function

Adaptation to device arrival - Newcomer centric

Newcomer Decision Problem

Choosing the computations (e_i, e_j binary variables):

$$\underset{e_i^k, e_j^k}{\textbf{minimize}} \sum_{(e_i,e_j)\in E^k} \textbf{com}(e_i, e_j)(e_i^k + e_j^k - 3 \cdot e_i^k \cdot e_j^k) \quad (11)$$

$$\textbf{subject to} \sum_{e_i \in V^k} \textbf{mem}(e_i) \cdot e_i^k \leq \textbf{cap}(a_k) \quad (12)$$

Solving the Newcomer Decision Problem

Must be solved on the newcomer

- It's an Integer quadratic program !
- It can be formulated as a Quadratic Knapsack Problem
- there are very good heuristics based on Dynamic Programming to solve QKP !
  - No optimality guarantees
  - but light enough for our devices

Adaptation to device removal

- We assume that agents detect the disappearance of any device a_k from the neighborhood
- We need to migrate the computations that where *hosted*, but not *owned* by a_k.
- Using the definition of neighborhood
  - $V_x[a_k]^- = V_x[a_k] \setminus \rho_x^{-1}(a_k)$, the variables involved
  - $V_f[a_k]^- = V_f[a_k] \setminus \rho_f^{-1}(a_k)$, the factors involved
  - $E[a_k]^- = E[a_k] \cap (V_x[a_k]^- \times V_f[a_k]^-)$, the edges involved

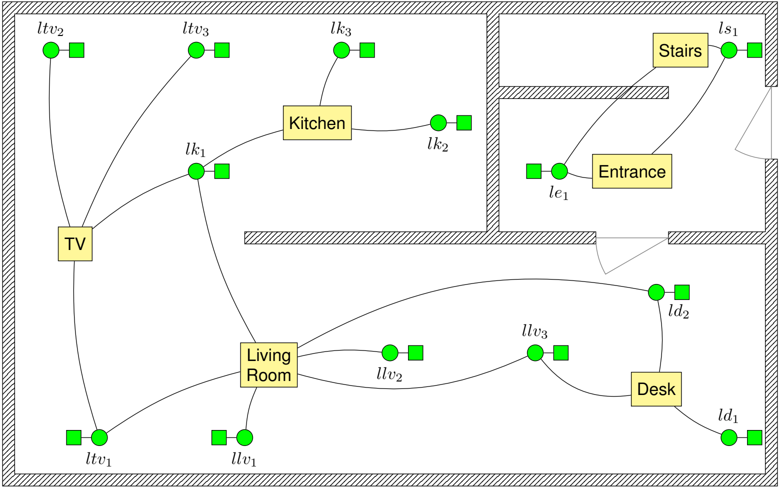
Adaptation to device removal

- Reuse the ILP for computation distribution
  - Restricting it to $V_x[a_k]^-$, $V_f[a_k]^-$, $E[a_k]^-$
  - Probably not optimal, but only requires local and limited knowledge of the SECP.
- Solving the reduced ILP
  - Smaller problem: could be distributed on the agents from the neighborhood

Experimental Setup

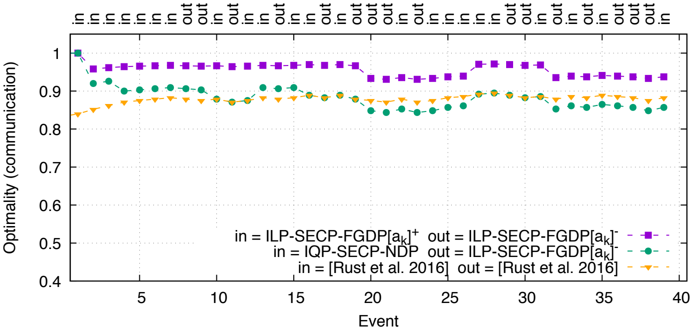
- Simulated smart home
- Two types of events :
  - device arrival : solved with the ILP and the QKP approaches
  - device removal : solved with the ILP approach
- The optimal distribution is also computed at each step
- We also compare the results with the (centralized) heuristic used in 2016
- Implementation
  - GLPK for ILP problems
  - Custom Dynamic Programing implementation for QKP

Simulated Smart home



First Experiment

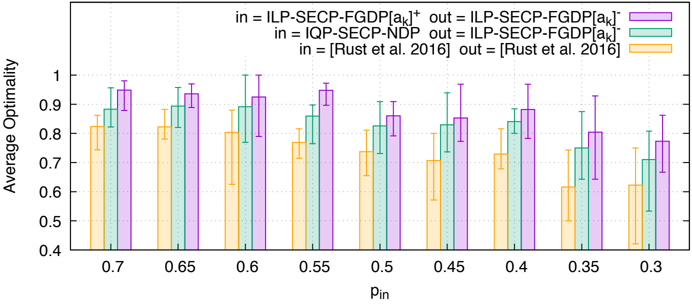
- Quality of the repaired distribution after each event.
- Removal events degrades the quality, but it's restored when adding devices.



Second Experiment

Influence of p_{in} on optimality

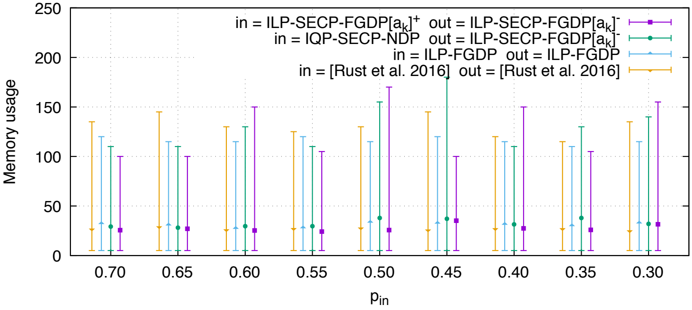
- Evaluates the robustness of the repair methods with more and more device removal.
- The higher p_{in}, the easier the adaptation is, since more devices are probably added.
- Average other 10 simulations of 20 events



Second Experiment

Influence of p_{in} on optimality

- Influence on memory usage (min, max and standard deviation)
- The approaches were not specifically designed to ensure a fair memory load share, yet we avoid excessive accumulation of computation on one device
- Average other 10 simulations of 20 events



Conclusions

Summary

- We discussed the problem of deploying factor graph elements within an open infrastructure composed of constrained devices.
- We proposed a model for an optimal deployment and several repair techniques to cope with device arrival and removal
- Experiments made on a simulated environment show that the proposed local and heuristic techniques have competitive optimality levels in comparison to restarting the deployment from scratch.
- These techniques only use limited and local knowledge and thus could be used in arbitrarily large systems.

Perspectives

Conclusions

Perspectives & future works

- When dealing with newcoming agents, how to choose which elements to propose ?
- Lighter methods for repairing the distribution.
- Distribute even the initial deployment process.