Déploiement de graphes de facteurs pour l'exécution d'algorithmes DCOP sur des infrastructures ouvertes

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

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

[2]Orange Labs

MINES Saint-Étienne

LABORATOIRE HUBERT CURIEN
UMR • CNRS • 5516 • SAINT-ETIENNE

CONNECTED INTELLIGENCE

orange™

Au programme

- Problème de configuration d'environnement intelligent (SECP)
- Problème du déploiement pour les DCOP et les SECP
- Déploiement en environnement dynamique
- Évaluation expérimentale
- Perspective

Problème de configuration d'environnement intelligent

Coordination décentralisée pour la maison intelligente

- Coordination directe entre les appareils connectés de la maison : pas de superviseur / coordinateur central
- Satisfaire les règles de l'utilisateur tout en minimisant la consommation énergétique
- Tous les calculs sont distribués directement sur les appareils connectés : ampoules, volets, etc.
- Des objets contraints
  - mémoire et puissance de calcul limitées
  - capacité de communication contrainte

SECP Model

Actuateurs :
Ampoules connectés, TV, volets, etc.

Senseurs :
Détecteur de présence, luminosité, etc.

Modèle de dépendance physique :
Modèle d'éclairage du salon

Préférences utilisateur :
sous forme de règles, donne un état cible de l'environnement ;

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

SECP Model

Actuateurs :

- Variable de décision x_i, Domain $\mathbf{x}_i \in \mathcal{D}_{x_i}$
- Fonction de coût$c_i : \mathcal{D}_{x_i} \rightarrow \mathbb{R}$

Senseurs :

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

Modèle de dépendance physique :

- Donne l'état théorique de l'environnement en fonction des variables d'actuateurs
- Variable y_j representant cet état
- Fonction $\phi_j : \prod_{\varsigma \in \sigma(\phi_j)} \mathcal{D}_\varsigma \rightarrow \mathcal{D}_{y_j}$

Préférences utilisateur :

- Fonction d'utilité u_k
- Distance entre l'état théorique et l'état cible de l'environnement from the current expected state to the target state of the environnement

Formulation du SECP en DCOP

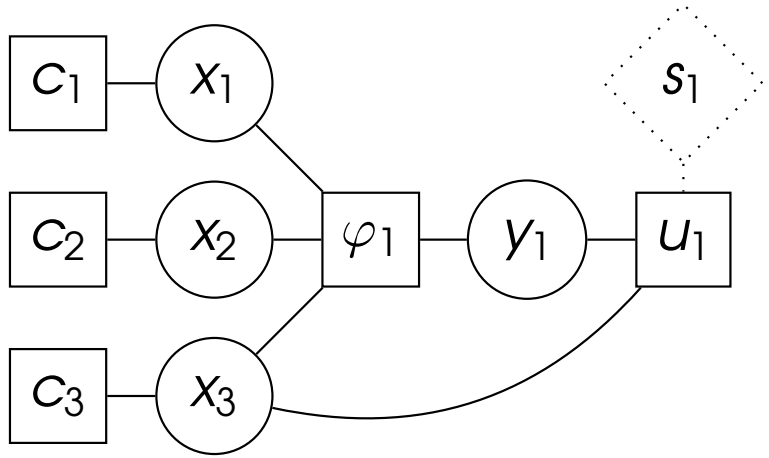
- Problème d'optimisation multi-objectifs, sous contraintes

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

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

- DCOP mono-objectif :

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



DCOP

Problème d'optimisation distribuée sous contraintes

Un DCOP est un tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mu \rangle$, where :

- $\mathcal{A} = \{a_1, \ldots, a_{|A|}\}$ un ensemble d'agents ;
- $\mathcal{X} = \{x_1, \ldots, x_n\}$ un ensemble de variables ;
- $\mathcal{D} = \{\mathcal{D}_{x_1}, \ldots, \mathcal{D}_{x_n}\}$ les domaines des variables x_i ;
- $\mathcal{C} = \{c_1, \ldots, c_m\}$ un ensemble de contraintes souples, où chaque c_i définit un coût $\in \mathbb{R} \cup \{\infty\}$ pour chaque combinaison d'affectation aux variables impliquées dans la contrainte ;
- μ une fonction associant chaque variable à un agent.

Une *solution* au DCOP est une affectation de valeur à toutes les variables qui minimise la somme totale des coûts $\sum_i c_i$.

La fonction de distribution

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

- fonction surjective, des variables vers les agents
- donne le contrôle de chaque variable x_i à un agent $\mu(x_i)$

Hypothèses courantes :

- chaque agent contrôle exactement une variable (bijection)
- que des contraintes binaires

Problèmes distribués réels :

- les agents sont exécutés sur des appareil réels
- l'ensemble d'appareils est une donnée du problème
- pas toujours de lien 'évident' entre une variable et un appareil

Problèmes réels

Modeliser des problèmes distribués réels

Un agent pour chaque variable :

- plusieurs agents pour un appareil
- comment décider quel appareil doit héberger chaque agent ?

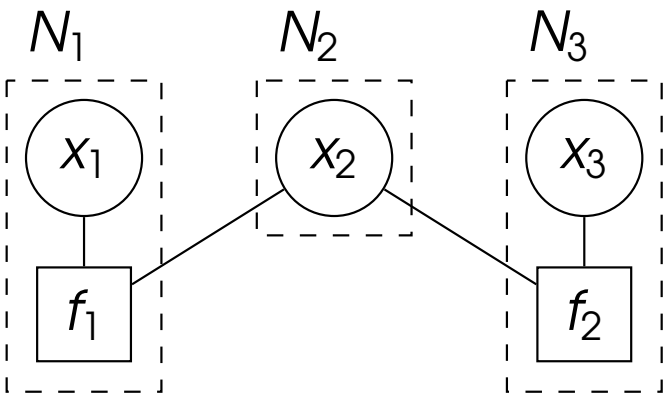
Un agent par appareil :

- un agent doit contrôler plusieurs variables
- comment décider quel agent est responsable de chaque variable ?

Algorithmes opérant sur les graphes de facteurs

Les facteurs doivent aussi être déployés

- un calcul pour chaque variable
- un calcul pour chaque contrainte (aka facteur)



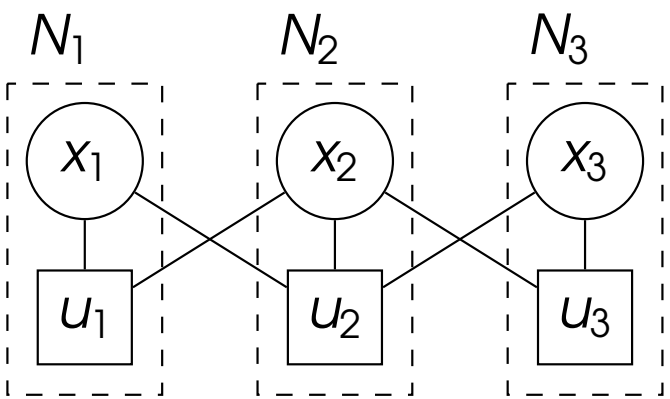
Comment décider quel agent doit héberger les calculs liés aux facteurs ?

Graphe de facteurs utilitaire

Max-Sum

Deux modélisations possibles :

- graphe de facteurs basés sur les interactions
- graphe de facteurs utilitaire



- Modélisation difficile
- Moins efficace : plus de facteurs, cycles, etc.
- Ne résout pas le problème des variables abstraites !

Problème de déploiement de graphe de facteurs

Pour la configuration d'environnements intelligent

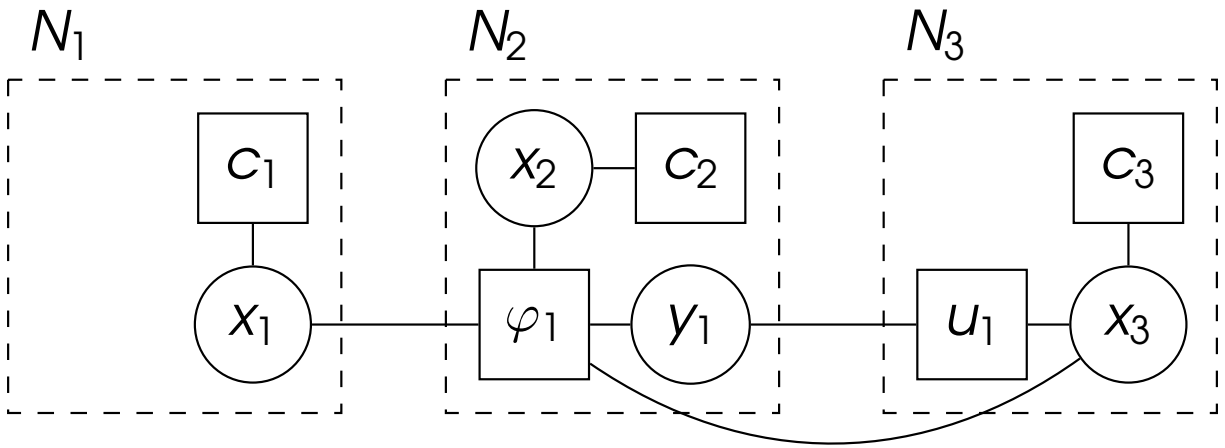
Le problème du déploiement :

- définir la fonction de déploiement μ.
- définir la notion de déploiement *optimal* : dépend du problème
- déploiement optimal $\equiv$ partitionnement de graphe : NP-hard !

Problème d'optimisation : la programmation linéaire en nombres entiers (pour le partitionnement de graphe).

Déployer le graphe de facteur du SECP

- Appareils à capacité mémoire limitée
- Communications chères et faible bande passante
- Héberger les variables liées à un actuateur sur ce dernier
- Objectif : **minimiser la communication entre les agents**



Programme linéaire binaire pour la distribution des calculs

- **com**(x_i, f_j) : charge de communication entre la variable x_i et le facteur f_j
- **mem**(e) : empreinte mémoire d'un calcul et **cap**(a_k) la capacité mémoire d'un appareil
- x_i^k et f_i^k : variables binaires liant les éléments du FG aux agents et pour linéariser $\alpha_{ijk} = x_i^k \cdot f_j^k$

Programme linéaire binaire

Déploiement des calculs d'un graphe de facteur

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

subject to

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

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

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

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

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

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

Programme linéaire binaire

Contraintes spécifiques au SECP

- fixer les variables d'actuateur et leur facteur de coût sur les appareils qui les possèdent
- contraintes supplémentaires pour respecter les capacités mémoires

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

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

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

Résolution du programme linéaire pour le déploiement

NP-hard, mais réaliste avec un algoritme branch-and-cut.

Mais ce n'est pas distribué !

- ça pourrait l'être : simplexe distribué
- probablement trop lourd pour les appareils de notre environnement
- utilisable lors de la phase d'initialisation du système
- fournit une référence pour l'optimalité, utile pour les comparaisons

le SECP est un problème dynamique

Infrastructure dynamique :

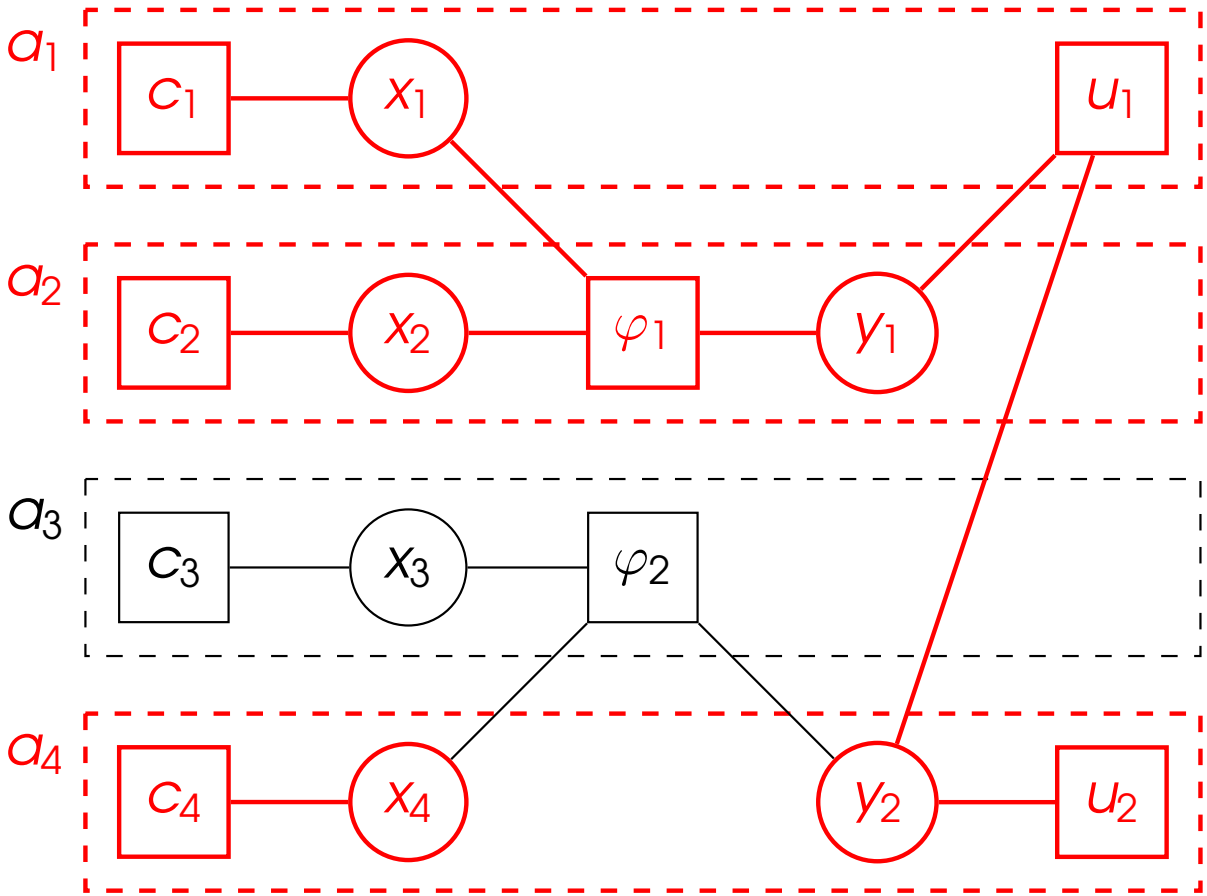
- Des appareils peuvent disparaitre
- De nouveau appareils peuvent être ajoutés

En cours de fonctionnement :

- Impossible de résoudre à nouveau le programme linéaire
- Il faut *réparer* le deploiement : auto-adaptation
- ne considérer qu'une sous-partie du graphe de facteur : le voisinage

Notion de voisinage

- Le *voisinage* d'un agent a_k est l'ensemble des agents qui hébergent un calcul lié à un calcul hébergé par a_k.
- Ensemble des arc du FG connectés aux agents du voisinage : $E[a_k]$
- Ensembles des variables et facteurs du voisinage : $V_x[a_k]$ et $V_f[a_k]$



Voisinage $A[a_2] = a_1, a_2$ et a_4
Ensembles associés $E[a_2]$, $V_x[a_2]$ et $V_f[a_2]$

Adaptation à l'arrivée d'agent - version ILP

- Réutiliser le programme linéaire binaire
  - Restreint au voisinage du nouvel agent
  - Sous-optimal, mais ne nécessite qu'une connaissance locale et limitée du système.

- Résoudre le programme linéaire restreint :
  - Problème plus petit : distribuable sur les agents du voisinage
  - Pire cas : le nouvel agent est connecté à tous les autres agents

Adaptation à l'arrivée d'agent - Appel à proposition

Problème du nouvel arrivant

- Problème du nouvel arrivant :
  - Le nouvel arrivant émet un appel à proposition pour déplacer certains calculs
  - Il choisit ensuite un ensemble de calcul en fonction des coût de communication et de sa propre capacité mémoire

- Chaque voisin $a_\ell \in \mathcal{A}[a_k]$ envoie une proposition $\langle V^{\ell \to k}, E^{\ell \to k}, \mathbf{com} \rangle$,
  - $V^{\ell \to k}$: calculs proposés
  - $E^{\ell \to k}$ les arc connectés à ces calculs
  - **com** la fonction de coût de communication

Adaptation à l'arrivée d'agent - Appel à proposition

Problème du nouvel arrivant

Choix des calculs (e_i, e_j variables binaires) :

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

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

Résoudre le problème du nouvel arrivant

Doit être résolu par le nouvel arrivant

- Programme quadratique
- On peut le formuler comme un problème de sac à dos quadratique (QKP)
- heuristique en programmation dynamique très efficace !
  - pas de garantie d'optimalité
  - assez léger pour nos appareils

Adaptation à la disparition d'agent

- Hypothèse : les agents détectent la disparition de tout appareil a_k de leur voisinage
- Il faut déplacer les calculs hébergés (mais non *possédés*) par a_k
- Avec la notion de voisinage
  - $V_x[a_k]^- = V_x[a_k] \setminus \rho_x^{-1}(a_k)$, les variables impactées
  - $V_f[a_k]^- = V_f[a_k] \setminus \rho_f^{-1}(a_k)$, les facteurs impactés
  - $E[a_k]^- = E[a_k] \cap (V_x[a_k]^- \times V_f[a_k]^-)$, les arc impactés

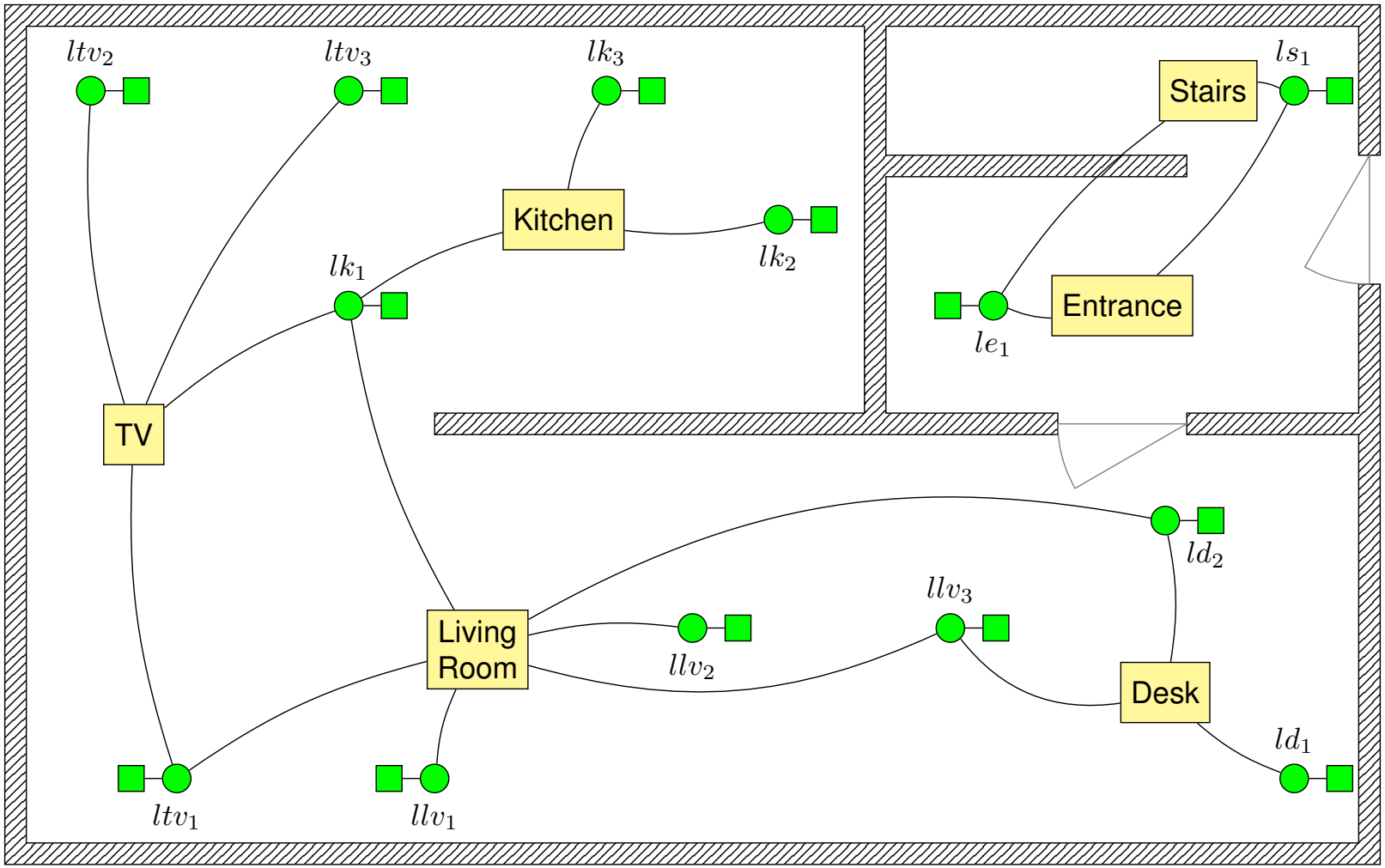
Adaptation à la disparition d'agent

- Réutilisation le programme linaire bianire pour la distribution
  - Restreint à $V_x[a_k]^-$, $V_f[a_k]^-$, $E[a_k]^-$
  - Sous-optimal, mais ne demande qu'une connaissance locale et limitée du système.
- Résoudre le programme linéaire restreint
  - problème plus petit : distribuable sur les agents du voisinage

Expérimentations

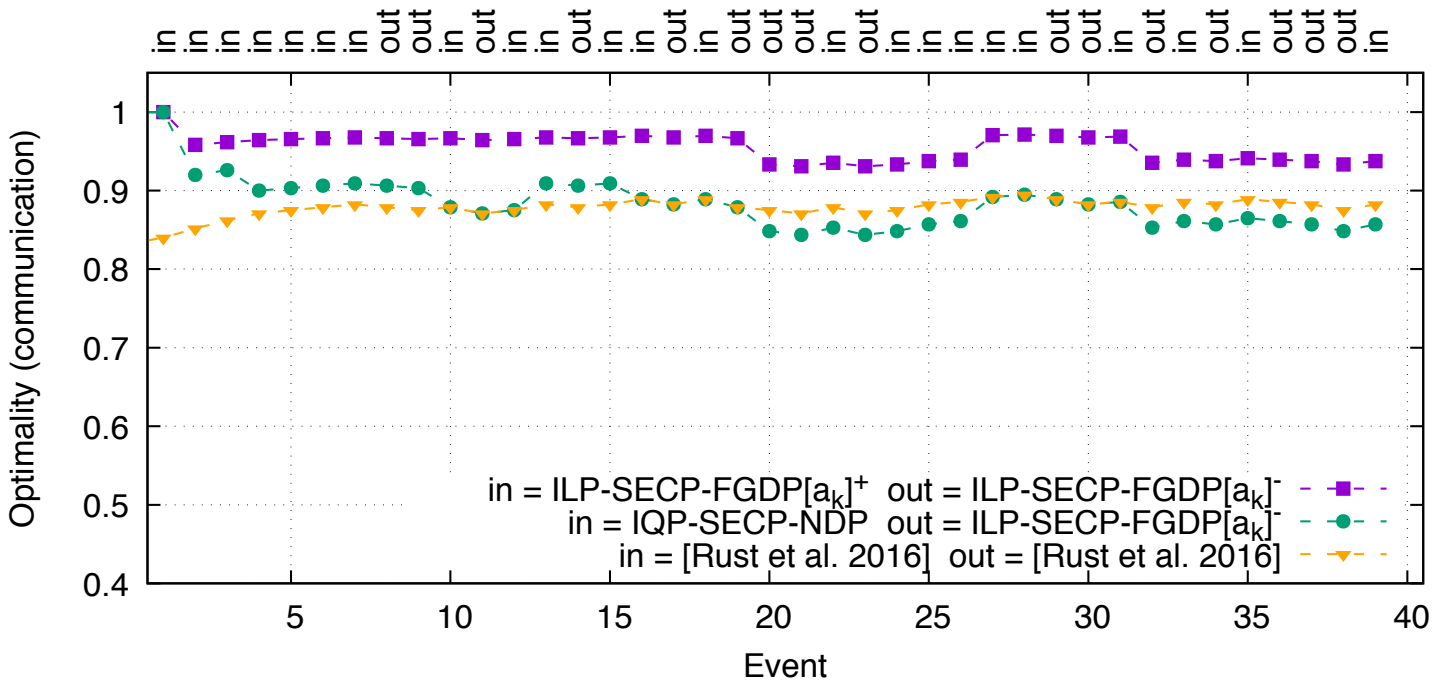
- Maison intelligente
- Deux types d'évènements
  - arrivée : avec les approches ILP et QKP
  - départ : avec l'approche ILP
- On calcule aussi la distribution optimale à chaque étape
- Nous comparons les résultats avec l'heuristique centralisée utilisée en 2106
- Implementation
  - GLPK for ILP problems
  - Implémentation en python pour QKP

Simulated Smart home



Suite d'évènement In / Out

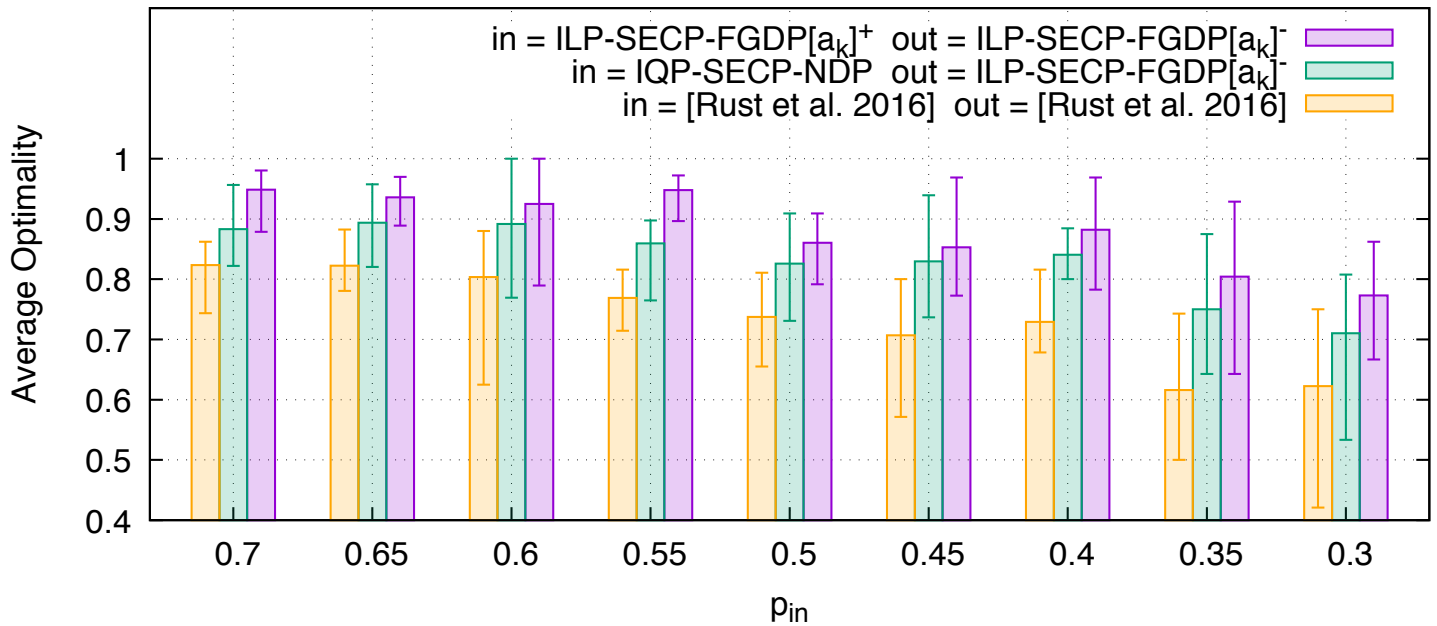
- Qualité de la distribution après chaque évènement.
- Les départ font baisser la qualité, mais les arrivées la restaure.



Robustesse du système

Influence de p_{in} sur l'optimalité

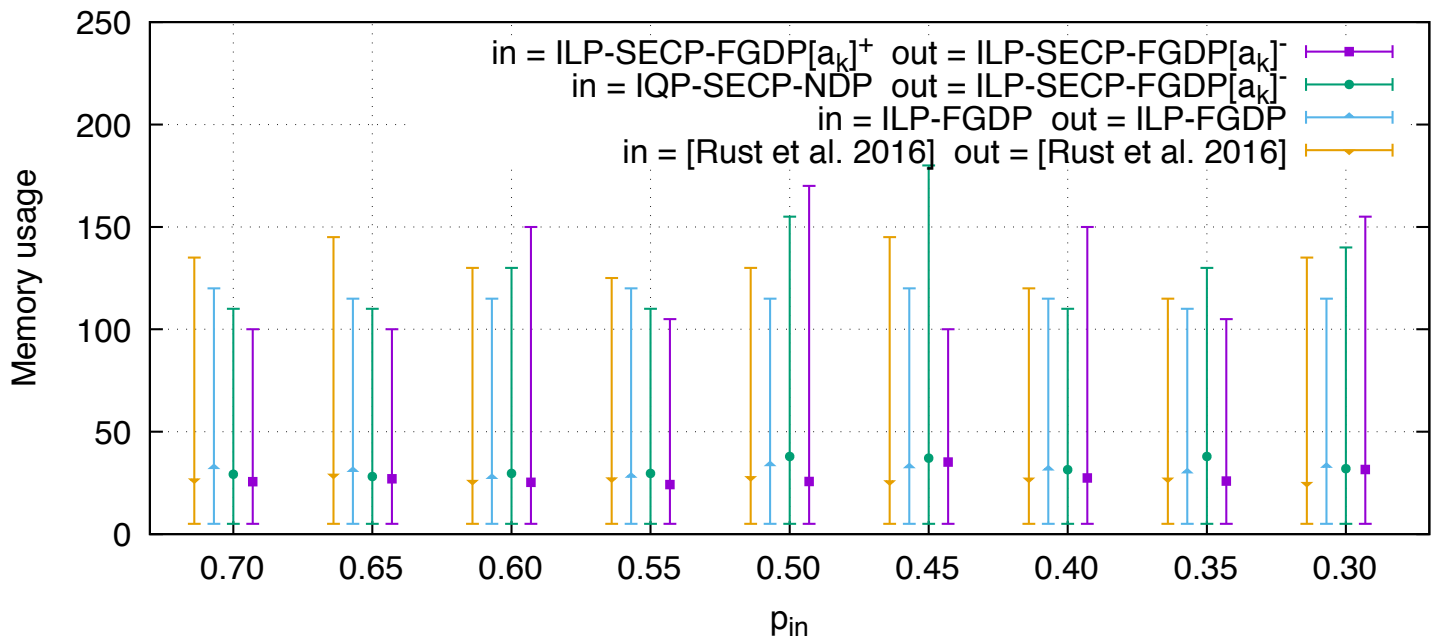
- Evaluer la robustesse des techniques de réparation avec de plus en plus de disparitions d'agents.
- Plus p_{in} est élevé, plus la réparation est aisée car plus d'équipements sont disponibles.
- Moyenne sur 10 simulations de 20 évènements



Robustesse du système

Influence de p_{in} sur l'optimalité

- Influence sur l'utilisation mémoire
- Ces techniques de réparation ne sont pas conçues pour une distribution équitable, mais évitent tout de même une concentration excessive
- Moyenne sur 10 simulations de 20 évènements



Conclusions

- Problème du déploiement d'un graphe de facteurs sur une infrastructure dynamique composée d'appareils contraints.
- Modèle de déploiement optimal et plusieurs techniques de réparation pour l'arrivée et le départ d'appareils.
- Expérimentation sur un environnement simulé : les réparations locales sont compétitives et réalistes.
- Ces techniques n'utilisent qu'une connaissance locale et limitée du système et pourraient donc être utilisées sur des systèmes arbitrairement grand.

Perspectives

- Lors de l'arrivée d'un agent, quel calcul proposer ?
- Trouver des méthodes de réparation plus légères.
- Déploiement initial en mode distribué.