Approche par optimisation distribuée pour la configuration autonome et spontanée d'environnements intelligents

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

[1] Orange Labs, France
{pierre.rust,fano.ramparany}@orange.com
[2] MINES Saint-Étienne, Institut Henri Fayol & Laboratoire Hubert Curien UMR CNRS 5516, France
gauthier.picard@emse.fr

Mots-clés : *internet des objets, optimisation distribuée, problème de configuration*

1 Le problème de configuration spontanée

Il est prévu que 25 milliards d'objets soient connectés à internet d'ici 2020. Ces objets sont très abordables mais assez contraints en terme de puissance de calcul et de mémoire. Nous nous intéressons ici à la coordination spontanée de tels objets dans le cadre de l'intelligence ambiante (comme dans la figure 1). Actuellement, les systèmes existants reposent sur des approches centralisées ou par *cloud*, ce qui peut représenter une limitation en terme de robustesse (point de rupture central) et de spontanéité du comportement exhibé (latence importante). Nous proposons ici de doter chaque objet, malgré sa puissance limitée, de capacités de coordination et d'optimisation distribuée pour que l'ensemble des objets atteigne une configuration optimale, au sens de (i) l'adéquation aux besoins exprimés par l'utilisateur et (ii) de la consommation électrique. Les besoins utilisateur correspondent à des règles de confort que le système doit mettre en œuvre automatiquement, comme par exemple « si une personne est présente dans le salon après 20h, atteindre un niveau de luminosité de 60% ». Contrairement aux approches du marché, l'utilisateur n'a pas à spécifier quels objets doivent être allumés/éteints pour exécuter cette règle. Les objets doivent se coordonner de manière spontanée, tout en minimisant la consommation électrique. Pour ce faire, nous proposons le modèle SECP (*Smart Environment Configuration Problem*) pour spécifier ce problème de configuration spontanée d'environnements intelligents [4].

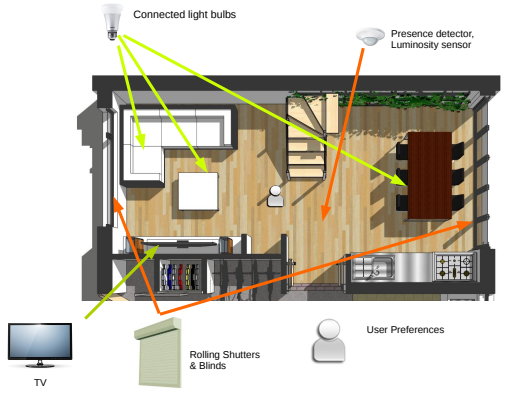

FIG. 1 – Un environnement intelligent à configurer

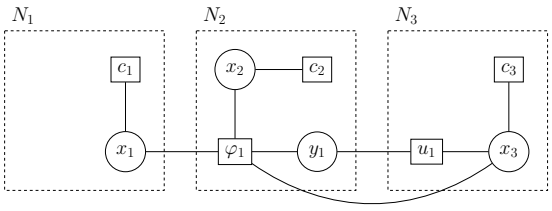

FIG. 2 – Déploiement d'un SECP sur les objets

2 Le modèle d'optimisation distribuée

Un modèle SECP peut être traduit en problème d'optimisation sous contraintes [4], qui peut être résolu de manière centralisée, ou distribuée en utilisant l'approche DCOP (*Distributed Constraint Optimization Problem*) par envoi de messages entre les objets concernés [1]. Ici les variables de décision sont les niveaux d'activation des objets (e.g. puissance d'une ampoule), ainsi que les propriétés physiques requises par l'utilisateur (e.g. niveau de luminosité dans le salon). Les contraintes représentent les règles utilisateurs, des modèles physiques (e.g. influence d'une ampoule sur la luminosité du salon) et les lois de consommation électrique. Le DCOP résultant peut être traduit en graphe de facteurs (graphe bipartite reliant des variables aux contraintes), dont les composants sont ensuite déployés sur les objets (cf. figure 2). Chaque objet prend des décisions sur des variables les concernant, sous contraintes connues par lui, ou par un voisin (auquel cas il devra interagir). Ce déploiement est effectué afin de minimiser ces interactions (par une approche heuristique pour plus de réactivité). Une fois les variables et facteurs déployés sur les objets (chaque fois que l'utilisateur rentre une nouvelle règle sur sa tablette), les objets mette en œuvre un protocole d'optimisation distribué. Plusieurs algorithmes existent pour résoudre des DCOP. Nous proposons ici d'utiliser des algorithmes d'inférence distribués, DPOP (Distributed Pseudotree Optimization Procedure) qui est optimal [3] et MaxSum qui est approché [2], car ils nécessitent en général moins de mémoire que les algorithmes de recherche distribués comme ADOPT (Asynchronous Distributed Optimization) [1].

3 Performances des algorithmes d'inférence distribués

Nous avons utilisé un simulateur d'environnement intelligents pour évaluer les performances de DPOP et MaxSum sur des instances ayant des modèles physiques de plus en plus complexes, et un nombre de règles de plus en plus grand, pour un configuration matérielle constante. Comme le montre la figure 3, nos expérimentations montrent que DPOP est légèrement plus rapide (mais sur simulateur, et non en contexte réellement distribué), alors que MaxSum génère beaucoup moins de messages tout en restant optimal dans 99.8% des cas. La charge réseau et le temps de réactivité dépend fortement du nombre de cycles dans le graphe de contrainte. MaxSum se montre un excellent compromis en terme d'optimalité, de temps de réponse et de charge réseau. Les résultats d'exécution d'ADOPT ne sont pas retranscrits ici, car l'algorithme ne termine pas en temps raisonnable sur les instances considérées.

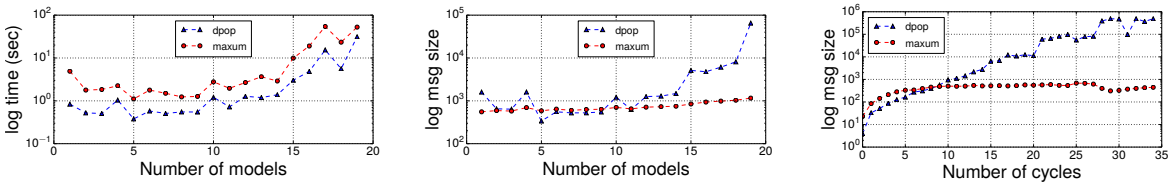

FIG. 3 – Performances de DPOP et MaxSum

Références

[1] J. Cerquides, A. Farinelli, P. Meseguer, and S. D. Ramchurn. A tutorial on optimization for multi-agent systems. *The Computer Journal*, 57(6) :799–824, 2014.

[2] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*, pages 639–646, 2008.

[3] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 266–271, 2005.

[4] Pierre Rust, Gauthier Picard, and Fano Ramparani. Using Message-Passing DCOP Algorithms to Solve Energy-Efficient Smart Environment Configuration Problems. In *International Joint Conference on Artificial Intelligence*, 2016.