Nested Monte-Carlo Search of Multi-agent Coalitions Mechanism with Constraints

Souhila Arib[1(✉)], Samir Aknine[2], and Tristan Cazenave[3]

[1] EISTI-Ecole Internationale des sciences du traitement de l'information, Avenue du Parc, 95000 Cergy, France
souhila.arib@eisti.fr
[2] LIRIS-Université Claude Bernard Lyon 1 - UCBL, 69622 Villeurbanne Cedex, France
samir.aknine@univ-lyon1.fr
[3] LAMSADE-Université Paris Dauphine Place du Mal de Lattre de Tassigny, 75775 Paris Cedex 16, France
cazenave@lamsade.dauphine.fr

Abstract. This paper develops and evaluates a coalition mechanism that enables agents to participate in concurrent tasks achievement in competitive situations in which agents have several constraints. Here we focus on situations in which the agents are self-interested and have not a priori knowledge about the preferences of their opponents, and they have to cooperate in order to reach their goals. All the agents have their specific constraints and this information is private. The agents negotiate for coalition formation (CF) over these constraints, that may be relaxed during negotiations. They start by exchanging their constraints and making proposals, which represent their acceptable solutions, until either an agreement is reached, or the negotiation terminates. We explore two techniques that ease the search of suitable coalitions: we use a constraint-based model and a heuristic search method. We describe a procedure that transforms these constraints into a structured graph on which the agents rely during their negotiations to generate a graph of feasible coalitions. This graph is therefore explored by a Nested Monte-Carlo search algorithm to generate the best coalitions and to minimize the negotiation time.

Keywords: Multi-agent systems · Coalition formation · Coordination · Negotiation

1 Introduction

Forming coalitions of agents which are able to effectively perform tasks is a key issue for many practical application contexts. This paper mainly focuses on self-interested agents which aim to form coalitions with other agents as they cannot reach their objectives individually. Several methods have been developed to control the behaviors of the agents involved in such process [12]. However few CF

A. Bikakis and X. Zheng (Eds.): MIWAI 2015, LNAI 9426, pp. 80–88, 2015.
DOI: 10.1007/978-3-319-26181-2_8

mechanisms cope with the dynamic of the constraints of the agents in such contexts. Indeed, these constraints can gradually be revealed, and relaxed by the agents at different moments of the negotiation in order to meet the requirements of their opponents and thus to ease the convergence. Some coalition methods have been developed to determine formerly to the negotiation the optimal coalitions and take into account the constraints of the agents involved in the coalition process. These methods have addressed important issues such as computational complexity and heuristics approaches for the optimal coalition structure generation, [7,10,13]. In this paper, we focus on contexts where agents neither have the same utility functions, nor they reveal these functions. Thus, it is infeasible to precisely estimate a priori the corresponding utility of each agent for each feasible proposal of solution with current optimal coalitions search algorithms. The issues with processing the constraints of the agents in the negotiation phase for the coalition formation deserve a particular attention and a deep study. Yet, only few works proposes a mechanism to deal with the dynamic of such constraints while agents negotiate them. Note that, since we consider that the agents are self-interested and do not share their information and computations, our aim is not to identify the optimal solution of the coalitions, but to ease the convergence to an agreed common solution for these agents. Our main contribution is a new mechanism that enables agents to negotiate and form coalitions. This mechanism is based on three main abstractions: a constraint graph, a coalition graph and Nested Monte-Carlo search method. First, we develop a constraints based graph which handles the revealed constraints of the agents. This graph of constraints can be used to specify different types of constraints relations, such as a constraints ordering over potential decision outcomes. Building upon this, we transform this representation into a flat representation of coalitions in the graph of coalitions. Each level of this graph allows generating a set of possible coalitions and in this set the agent selects the best coalitions that can be accepted. This graphical representation of constraints and coalitions specifies constraints relations in a relatively compact, intuitive, and structured manner. To explore this graph of coalitions, we first define the problem and link it to other existing problems, so that approximate solution techniques and anytime heuristics that provide increasingly better solutions if given more time can be re-used. We advise new solutions that allow agents use a nested Monte-Carlo search algorithm [1] which finds the best coalition that maximizes the utility of each agent. Nested Monte-Carlo search methods address the problem of guiding the search towards better states when there is no available heuristic. These methods use nested levels of random games in order to guide the search of coalitions. These algorithms have been studied theoretically on simple abstract problems and applied successfully to several games [4]. Specifically, this paper advances the state of the art in the following ways. We advise new anytime heuristics to find approximate solutions fast, we empirically evaluate our algorithm and show that it computes (in less than 600 milliseconds) 689 proposals of solutions for non-trivial problems involving up to 30 agents and 50 tasks. Thus, our work encompasses essential aspects of the coalition formation, from the coalition model, negotiation, and an

anytime heuristic. The reminder of the paper is organized as follows. Section 2 briefly describes the related works. Section 3 introduces some preliminaries and the case study. Section 4 presents the coalition formation mechanism, and a final section will conclude the work with a summary of the contributions.

2 Related Work

In game-theoretic perspective, coalitional games with constraints have been addressed by a number of works. However, none of these structures is able to model agents' negotiations for reaching joint agreements. [3] proposes a game-theoretical study and focuses on strategic, core-related issues rather than computational analysis of the coalition formation. This work is more close to [7] where authors propose a constrained coalition formation model and an algorithm for optimal coalition structure generation. They develop a procedure that transforms the specified set of constraints, making it possible to identify all the feasible coalitions. Building upon this, they provide an algorithm for optimal coalition structure generation. [13] address the problem of coalition formation with sparse synergies where the set of feasible coalitions is constrained by the edges of a graph. Their aim is to check whether knowledge of the topology of an underlying social or organizational context graph could be used to speed up coalition enumeration and structure generation. [10] define the problem of allocating coalitions of agents to spatially distributed tasks with workloads and deadlines so as to maximize the total number of tasks completed over time. Nevertheless, these works have not deeply addressed the constraints of the agents in the proposed models or specify how agents negotiate over them to reach agreements. Constraints on coalition sizes have been considered for coalition value calculation [8,9,11]. However, the semantics of these constraints has not been used on the same level as it is done in this paper. [2,5] develop succinct and expressive representations for coalitional games. Such formalism could be used to encode the constraints, but this is not the main concern of the constrained CF mechanism considered in this paper.

3 Preliminaries and Case Study

To illustrate the coalition formation mechanism we propose, let us consider a carpooling example, where some travellers want to move from a city to another, and they want to share their means of transportation. Each traveller formulates to his agent the goals to be achieved. For example "I want to go from NY to Boston", his constraints as departure time, duration of the travel, and unit price of seat. To solve this problem, the agents have to deal with all the constraints and preferences over those of their associated travellers in order to enable them to share transportation. Agents negotiate for the coalitions to form to decrease the unit price of seat, increase the number of passengers, etc. They can step aside in favor of other agents, if an agreement can be found. More formally, consider a set of agents $\mathcal{N} = \{a_1, a_2, ..., a_n\}$, a set of actions $\mathcal{A} = \{b_1, b_2, ..., b_m\}$

and a set of constraints $\mathcal{C}_t = \{c_{t1}, c_{t2}, ..., c_{tk}\}$. The agents of $\mathcal{N}$ need to execute the actions of $\mathcal{A}$ by satisfying the constraints in $\mathcal{C}_t$. The constraints are defined as intervals, for instance: departure time: $D \in [10a.m., 12a.m.]$, travel duration in hours: $T \in [1H, 2H]$ and price: $P \in [20, 25]$. The agents' preferences are represented using a preference relation $\succ$ for those they want to share a car with, for instance $a_x \succ_i a_y$ (for agent a_i, a_x is preferred to a_y). We consider a coalition c as a nonempty subset of $\mathcal{N}$ ($c \subseteq \mathcal{N}$). We define $\mathcal{C}$ as the set of all possible coalitions. For a coalition c to be formed, each agent a_i in c should get a certain satisfaction. This satisfaction is defined by a utility function $u_i : \mathcal{C} \mapsto \mathbb{R}$. Note that a coalition is acceptable for agent a_i if it is preferred over, or equivalent to a reference coalition, $u_i(ref)$, which corresponds to the minimal guaranteed gain of the agent during the negotiation. A solution of the negotiation for each agent a_i introduces a coalition structure denoted CS_i which is defined on $\mathcal{N}$ with its associated utility $u_i(CS_i)$. CS_i contains a set of coalitions $\{c_1, c_2, .., c_q\}$ to be formed for the set of actions $\mathcal{A}_i \subseteq \mathcal{A}$ where a_i is involved. Furthermore, for every $q' \in [1, q], c_{q'} \subseteq \mathcal{N}$ and $c_{q'}$ performs a set of actions $\mathcal{A}_{c_{q'}} \subseteq \mathcal{A}$ and $\forall (x, y) \in [1, q]^2, x \neq y$, $\mathcal{A}_{c_x} \cap \mathcal{A}_{c_y} = \varnothing, \bigcup_{q'=1,..,q} \mathcal{A}_{c_{q'}} \subseteq \mathcal{A}$ and $\bigcup_{q'=1,..,q} c_{q'} \subseteq \mathcal{N}$. The set of all coalition structures is denoted $\mathcal{S}$.

4 Coalition Formation Mechanism (CFM)

In order to satisfy the goals they have to achieve, the agents perform negotiations on the coalitions they want to form. So, the CFM requires an analysis step of constraints that agents exchange in order to guide the choice of the coalitions and a step of generating coalition structures from these constraints. Constraint analysis relies on constructing a graph of constraints and coalition generation is based on the mapping of the constraints to possible coalitions in a coalition graph. Exploring the search graph of coalitions toward better states is based on a Nested Monte-Carlo algorithm.

4.1 Constraint Graph

An effective technique for solving a coalition formation problem is a heuristic search through abstract problem spaces. The first problem space can be represented by a directed connected graph, where nodes correspond to constraint sets and edges correspond to actions (cf. Fig. 1). The constraint graph may include many paths from the start to any node. Since the agents are self-interested, to search among the constraints to deal with in the coalitions, every agent constructs its own graph of constraints based on its own constraints and those revealed by other agents during this negotiation. Given a set of constraints that must be satisfied by an agent to execute a set of actions and starting from the source node labeled with $\{b\colon \varnothing,\ c_t\colon \varnothing\}$, initially there are not constraints and actions associated with this source node, let us define a graph denoted $G(c_t, b)$ as follows.

Definition 1. *Given a node labeled* $\{b: \varnothing, c_t: \varnothing\}$ *the constraint graph* $G(c_t, b)$ *is a directed connected graph, containing all possible nodes of constraints represented by intervals, labeled* $\{X_1, ..., X'_k\}$ *for each action* $b_i, 1 \leq i \leq m$*, that has to be executed by the agent. Each node has a utility labeled* u*, and directed edges from this node are labeled* $\{b_j, ..., b_k\}$ *where* $1 \leq j \ .. \ k \leq m$.

A constraint graph gathers, the most preferred constraints' intervals in its nodes. At the root node, no action and constraint are added. Each node generates a finite set of child nodes which correspond to the accepted sets of constraints, where the first node of the graph is an outgoing node and the last nodes are incoming nodes. This constraint graph is built following a preference rate on the intervals of constraints.

Let us consider two agents, a_i which has its own interval X and receives from a_j an interval Y. The agent a_i wants to create a new interval Z that meets its constraints and those of a_j, $\{Z \models a_i\}$, by merging its interval and the one received from a_j. We will adopt the convention of the left and right endpoints of an interval X by $\underline{X}$ and $\overline{X}$, respectively [6]. First, a_i tests if $X \subseteq Y$. Thus, if $\underline{Y} \leqslant \underline{X}$ and $\overline{X} \leqslant \overline{Y}$, it will get $X \subseteq Y$ and $Z = X$. Else the agent tests if $X \cap Y$ and calculates the new interval Z. If $X \cap Y = \varnothing$ there are no points in common with a_j. Otherwise, $Z = \{max\{\underline{X}, \underline{Y}\}, \ min\{\overline{X}, \overline{Y}\}\}$ and tests if it complies with its actions. If a_i does not choose this interval, it calculates $Z = X \cup Y$ which means the union X and Y and tests if it complies with its actions. For more details about the operations over the intervals see [6]. Based on this graph, constraint analysis consists for an agent of comparing and grouping its constraints and those received from others. A natural constraint graph analysis involves constructing and linking optimal nodes. Constraints are gathered based on their relations into sets represented in the nodes of this graph. Each level of the graph of constraints refers to an action to be performed by a coalition. The advantage of the suggested method consists in directing the search of the solutions of coalitions towards primary constraints, i.e., important constraints to satisfy, thus, reducing search complexity. To move from one node of this graph to another, an action is added to the graph. The utility of a move, which labels the corresponding edge in the search space, is the utility of the action when it is added and performed by the coalition. A solution path represents a particular succession order of the added actions, and the width of that order is the sum of the edge utilities on the solution path.

Let us consider constructing the constraint graph by the agent a_1 on our previous example using. First, assume that agent a_1 started a negotiation with agents a_2 to a_5 and in which each of these five agents revealed certain of its constraints. The actions that have to be executed are: b_1, b_2, b_3 which correspond respectively to: go from NY to Amherst, find a hotel room in Amherst, and go from Amherst to Boston. The constraints identified by a_1 for the action b_1 are: $D \in [10a.m., 01p.m.]$, $T \in [1H, 2H]$ and $P \in [20, 25]$ and for b_2 are: $D \in [01p.m., 02p.m.]$, $T \in [1H, 2H]$ and $P \in [20, 25]$. The nodes in the first level of the graph assemble possible sets of constraints concerning the action b_1. a_1

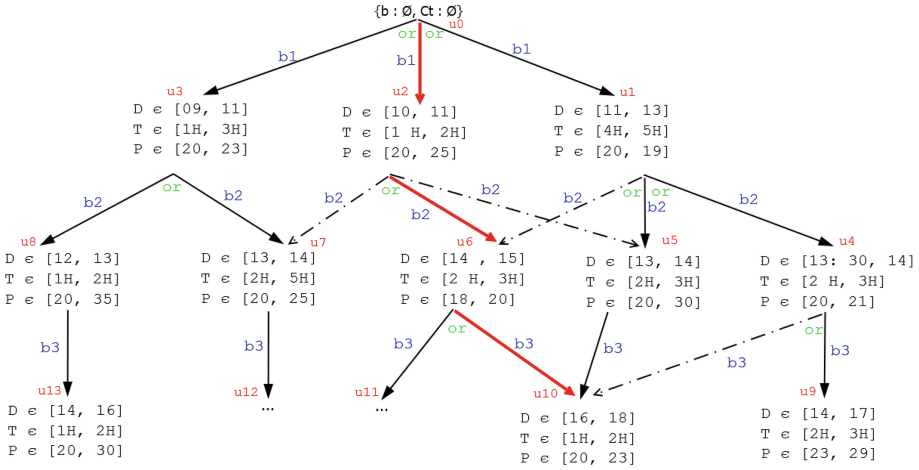

Fig. 1. An example of a graph of constraints against different actions of the agent a_1. Each node is labeled with its associated utility.

compares its own constraints and those received from these agents and creates new intervals of constraints X_{kc_t}.

Let us consider again the agent a_1 who received these intervals of constraints from the agent a_2 concerning the action b_1: $D \in [10a.m., 11a.m.]$, $T \in [1H, 3H]$ and $P \in [20, 35]$. So, $D \in [10a.m., 11a.m.] \subseteq [10a.m., 01p.m.]$ so $Z = [10a.m., 11a.m.] \models (a_1, a_2)$. Thus, a_1 selects the interval Z. For $T \in [1H, 2H] \subseteq [1H, 3H]$, $Z = [1H, 2H] \models (a_1, a_2)$, so a_1 chooses $T \in [1H, 2H]$. For $P \in [20, 25] \subseteq [20, 35]$, $Z = [20, 25] \models (a_1, a_2)$. These results are resumed in the Fig. 1. On the left of this figure, agent a_1 represents the first node created for the action b_1 by the ordered set of intervals $D \in [09a.m., 11a.m.]$, $T \in [1H, 3H]$ and $P \in [20, 23]$. We notice in this node that a_1 chooses the interval $[09a.m., 11a.m.]$ even if its departure time is not completely included in $D \in [10a.m., 01p.m.]$ because it has a good proposal of the seat price. $D \in [10a.m., 11a.m.]$, $T \in [1H, 2H]$ and $P \in [20, 25]$ are associated with the second one and $D \in [11a.m., 01p.m.]$, $T \in [4H, 5H]$ and $P \in [20, 19]$ with the last child. So, for each action, a_1 generates the different possible intervals of constraints that satisfy its action b_1. We observe that the constraints in each child node are created taking into account the end of execution of the antecedent action. So, to generate intervals of constraints that satisfy the action b_{i+1}, the agent takes into account the end time of b_i. This allows the agent to manage the relations between the actions that have to be executed. In this example, in the second level of the graph the agent a_1 identified for the action b_2 these intervals: $D \in [01p.m., 02p.m.]$, $T \in [2H, 5H]$ and $P \in [20, 25]$. The beginning of b_2 is in $[01p.m., 02p.m.]$ because b_1 ends at the latest at $01p.m.$. The dashed arcs show that nodes can share the same child nodes and the red and bold ones show the most preferred path from the root node to the last one, they result from the

Monte-Carlo exploration (detailed below). Every agent a_i which has to negotiate to execute an action b_i while satisfying its constraints c_t, chooses intervals of constraints: $X_{kc_t}(b_i) \models a_i$. It then creates child nodes for the feasible intervals that satisfy b_i. Each node created can be split under appropriate restrictions to other child nodes. The agent a_i starts with a node, labeled X_{kc_t}, for the action b_i. For each action b_{i+1} that must be executed after b_i and need negotiation, a_i creates the new intervals for $b_{i+1} : (X'_{1c_t}, ..., X'_{k'c_t})$, and splits the node b_i, X_{kc_t} to the child nodes $b_{i+1} : (X'_{1c_t}, ..., X'_{k'c_t})$. The agent a_i uses this procedure until no action need negotiation.

A notable detail of the constraints search space construction is that a solution is measured by its maximum path utility. We use an additive utility function, where a path is evaluated by summing its edge utilities. For each iteration, feasible solutions are only explored if their utility is not under a certain reference situation $u_i(ref)$. If an iteration is completed without finding a new possible solution, then all solutions provide less utility than that of the reference situation, $u_i(ref)$, thus, $u_i(ref)$ may be decreased and the search is repeated. The section below explains how a Nested Monte-Carlo algorithm explores a graph of constraints.

4.2 Constraint Graph Exploration with Nested Monte-Carlo Algorithm (NMC)

To optimize the search time for the new coalitions to propose or to accept, agents use the Nested Monte-Carlo algorithm. Nested Monte-Carlo Search is used for problems that do not have good heuristics. It was shown that memorizing the best sequence improves the mean result of the search. Experiments on different games gave very good results, finding a new world record of 80 moves at Morpion Solitaire, improving on previous algorithms at SameGame, and being more than 200,000 times faster than depth-first search at 16x16 Sudoku modeled as a Constraint Satisfaction Problem [1]. In the first step of the mechanism, the NMC explores each level of the graph of constraints and stores the best path of constraints that satisfies the agent a_i. The idea of NMC is to use lower sequences of simulations in order to decide the utility that an agent gets from a path at the current sequence. This step is necessary because agents do not have a priori a knowledge about the utility functions of the others. When all simulations of the underlying sequences have been performed, the agent utility is memorized in the best sequence; it means that it is possible to get the best path. The solution improves monotonically since our algorithm keeps track of the best proposal of solution found so far. Nested Monte-Carlo search combines nested calls with randomness in the playouts and memorization of the best sequence of moves. In nested rollouts, the rollouts are based on a heuristic. It implies that nested rollouts always improve on rollouts and on simply following the heuristic. When the base level does not use a heuristic but random moves, it is possible that a nested search gives worse results than a lower level search. It is useful to memorize the best sequence found so far in order to follow it when the randomized searches give worse results than the best sequence. The basic *sample* function

Algorithm 1. Nested play

```
1  begin
2      Require:nested (Position, level in the graph);
3      best utility = u(ref);
4      while not end of the graph do
5          if level is 1 then
6              move = argmax_move(sample (Play(Position, move)));
7          end
8          else
9              move = argmax_move(nested (Play(Position, move), level -1))
10         end
11         if utility of a move > best utility then
12             best utility = utility of the move;
13             bestSequence = Sequence after move;
14             bestMove = move of the sequence ;
15         end
16         Position = Play(Position, bestMove);
17     end
18     return best utility
19 end
```

(cf. Algorithm 1) just explores a graph randomly from a given position in the graph, agents use the function $play(position, move)$ which plays the move in the position and returns the resulting position. If none of the moves improves on the best sequence, the move of the best sequence is played, otherwise the best sequence is updated with the newly found sequence and the best move is played.

5 Conclusion

This paper has introduced a new coalition formation mechanism enriched with several principles to deal with the constraints of the agents and a Nested Monte-Carlo based search algorithm. We have detailed how the constraints are modeled as a graph and how this graph is explored using the Nested Monte-Carlo search. From the graph of constraints, each agent gets its most preferred path of constraints and constructs a coalition graph that is used to generate the coalitions to negotiate.

References

1. Cazenave, T.: Nested monte-carlo search. In: IJCAI, pp. 456–461 (2009)
2. Conitzer, V., Sandholm, T.: Complexity of constructing solutions in the core based on synergies among coalitions. Artif. Intell. **170**, 607–619 (2006)
3. Demange, G.: The strategy structure of some coalition formation games. Games Econ. Behav. **64**, 83–104 (2009)
4. Gelly, S., Silver, D.: Combining online and offline knowledge in UCT. In: ICML 2007, pp. 273–280 (2007)
5. Ieong, S., Shoham, Y.: Marginal contribution nets: A compact representation scheme for coalitional games. In: Proceedings of the 6th ACM Conference on Electronic Commerce, pp. 193–202. ACM (2005)

6. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. SIAM, Philadelphia (2009)
7. Rahwan, T., Michalak, T., Elkind, E., Faliszewski, P., Sroka, J., Wooldridge, M., Jennings, N.R.: Constrained coalition formation. In: AAAI (2011)
8. Rahwan, T., Ramchurn, S.D., Dang, V.D., Jennings, N.R.: Near-optimal anytime coalition structure generation. In: IJCAI, pp. 2365–2371 (2007)
9. Rahwan, T., Ramchurn, S.D., Dang, V.D., Jennings, N.R., Giovannucci, A.: An anytime algorithm for optimal coalition structure generation. J. Artif. Int. Res. **34**(1), 521–567 (2009)
10. Ramchurn, S.D., Polukarov, M., Farinelli, A., Truong, C., Jennings, N.R.: Coalition formation with spatial and temporal constraints. In: AAMAS, pp. 1181–1188 (2010)
11. Sandholm, T., Larson, K., Andersson, M., Shehory, O., Tohme, F.: Coalition structure generation with worst case guarantees. Artif. Intell. **111**, 209–238 (1999)
12. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artif. Intell. **101**, 165–200 (1998)
13. Voice, T., Ramchurn, S., Jennings, N.: On coalition formation with sparse synergies. In: AAMAS, pp. 223–230 (2012)