Multi-Criteria Coalition Formation Games

Ayumi Igarashi (✉)[1] and Diederik M. Roijers [1,2]

[1] Department of Computer Science, University of Oxford, Oxford, U.K.
[2] Artificial Intelligence Laboratory, Vrije Universiteit Brussel, Brussels, Belgium
ayumi.igarashi@cs.ox.ac.uk, droijers@ai.vub.ac.be

Abstract. When forming coalitions, agents have different utilities per coalition. Game-theoretic approaches typically assume that the scalar utility for each agent for each coalition is public information. However, we argue that this is not a realistic assumption, as agents may not want to divulge this information or are even incapable of expressing it. To mitigate this, we propose the *multi-criteria coalition formation game* model, in which there are different publicly available quality metrics (corresponding to different criteria) for which a value is publicly available for each coalition. The agents have private utility functions that determine their preferences with respect to these criteria, and thus also with respect to the different coalitions. Assuming that we can ask agents to compare two coalitions, we propose a heuristic (best response) algorithm for finding stable partitions in MC2FGs: *local stability search (LSS)*. We show that while theoretically individually stable partitions need not exist in MC2FGs in general, empirically stable partitions can be found. Furthermore, we show that we can find individually stable partitions after asking only a small number of comparisons, which is highly important for applying this model in practice.

Keywords: Multi-criteria, hedonic games, coalition formation games, preference elicitation, local search

1 Introduction

Coalitions are an essential part of life; typically we are not able to achieve our goals, or that of an organisation we are part of, by ourselves. Therefore, we need to cooperate in order to achieve these goals. For example, this paper has been written by a coalition of authors, who could not have written this paper individually, without the help of the other authors. *Hedonic games*, initiated by Banerjee et al. [4] and Bogomolnaia and Jackson [5], offer a versatile framework to model how coalitions are formed by multiple autonomous agents. An important consideration in this respect is *coalitional/individual stability*, i.e., no individuals wish to deviate from their current situation.

In order to determine which coalition structures are stable, we need to know the *preferences* over different possible coalitions for each agent. The standard setting of hedonic games [6] assumes that we know, or can compute, the exact preferences of each agent over possible coalitions. However, this is not a realistic assumption; not only may agents not want to divulge this information for social or privacy reasons, but the agents might not even be able to specify their utilities a priori to begin with. For example, when working for a company, it may not be socially acceptable to order different possible teams you may be on.

1.1 Our contributions

In this paper, we aim to mitigate this lack of information in two important ways. First, we take the information we do have into account, in the form of different quality metrics for coalitions. E.g., a manager who wants to assign teams, may rely on statistics from previous teams, such as a previous productivity metric, a number of occurred conflicts, and reports by previous team leaders on how creative teams have been. Combining this with a predictor for new possible coalitions, we may assign an expectation of how well such a team will do with respect to these criteria. This leads us to the formulation of a *multi-criteria coalition formation game (MC2FG)*, i.e., we have access to the quality of a coalition with respect to multiple quality criteria.

Contrary to previous work on related (vector-valued) cooperative games [20], we take a *utility-based approach* [16]: we assume that each agent, i, has a private utility function, $f^{(i)}$, that takes the multi-criteria value of a coalition, and produces a scalar utility. Because these functions are private, we cannot use these to check whether a proposed coalition is stable. We are however able to make some assumptions about $f^{(i)}$, e.g., that it is a monotonically increasing function w.r.t. all criteria. Furthermore, we assume that we can increase our knowledge about $f^{(i)}$, i.e., impose extra constraints, by asking agent i to compare two coalitions.

We make the following contributions. First, in Section 2, we propose our model: the *multi-criteria coalition formation game (MC2FG)*. We prove that while for single-criterion coalition formations individually stable partitions exist, this is not always the case for MC2FGs (Section 3). Therefore, we define a new heuristic search algorithm called *local stability search (LSS)* to investigate whether we can find stable partitions in practice (Section 4). We test this on random MC2FGs, as well as a class of MC2FGs we call $\texttt{Author} \times \texttt{Author}$, inspired on forming teams of scientists to work together. We show empirically that it is possible to find individually stable, and sometimes even Nash-stable, partitions. Further, we show that it is possible to limit the number of questions we need to ask to an agent to a number that is likely to be feasible to ask from human decision makers. We therefore conclude that this type of model, with individual utility functions but public quality metrics, is a promising area that warrants further research.

1.2 Related work

In the existing literature of cooperative games, Fernández et al. [8] were the first to incorporate multi-criteria characteristic functions. In [8], the classical solution concepts, such as the core, have been adapted to the multi-criteria setting (see also the survey by Tanino [20]). Faliszewski et al. [7] also extended weighted voting games to the multi-criteria cases in which coalitions are defined to be winning if they are winning in several levels of games specified by the propositional formula. We note that in these settings, the players' utilities are *transferable*; hence, the goal of these models is to divide the resulting values among players in a reasonable way. In contrast, our work assumes that the utilities are *non-transferable*, and focuses on coalition formation among players.

There is a rich body of the literature on preference elicitation in computational social choice. In particular, Balcan et al. [2] studyied the PAC (probably approximately correct) learnability of cooperative transferable utility games. Specifically, they investigate whether given random samples of coalitions, one can efficiently predict the unknown

values of coalitions as well as an allocation that is likely to be stable. Although there are some classes of games that are computationally intractable to learn, it has been shown that one can always stabilize a game by finding an allocation that is likely to be in the core. The most closely related to our approach is perhaps the paper by Benabbou and Perny [3], who designed an incremental preference elicitation procedure for the knapsack problem with multiple decision-makers; in their work, players have individual utilities for each item, and each item corresponds to its own criterion (in our framework).

2 The General framework

For $k \in \mathbb{N}$, let $[k] = \{1, 2, \ldots, k\}$. We define multi-criteria coalition formation games.

Definition 1. *A* multi-criteria coalition formation game *(MC2FG) is a triple $(N, \boldsymbol{q}, (f^{(i)})_{i \in N})$ where $N = [n]$ is a finite set of* players*, $\boldsymbol{q} : 2^N \to \mathbb{R}^m$ is a multi-valued set function that represents the quality $q_k(S)$ of a subset S for each criterion $k \in [m]$, and each $f^{(i)} : \mathbb{R}^m \to \mathbb{R}$ is a* private utility function *for each player $i \in N$. We assume that each $f^{(i)}$ is* strictly monotone*, namely, $f^{(i)}(\boldsymbol{x}) > f^{(i)}(\boldsymbol{y})$ whenever $\boldsymbol{x} > \boldsymbol{y}$.*

Throughout the paper, we assume that $\boldsymbol{q}(\emptyset) = \mathbf{0}$. We refer the subsets of N as *coalitions*. We let $\mathcal{N}_i$ denote the collection of all possible coalitions containing i. Preference relations derived from scalarisations can be naturally defined as follows. Let $i \in N$ and coalitions $S, T \in \mathcal{N}_i$. We say that player i *weakly prefers* S to T (denoted by $S \succeq_{f^{(i)}} T$) if $f^{(i)}(\boldsymbol{q}(S)) \geq f^{(i)}(\boldsymbol{q}(T))$; player i *strictly prefers* S to T (denoted by $S \succ_{f^{(i)}} T$) if $f^{(i)}(\boldsymbol{q}(S)) > f^{(i)}(\boldsymbol{q}(T))$; player i is *indifferent* between S and T (denoted by $S \sim_{f^{(i)}} T$) if $f^{(i)}(\boldsymbol{q}(S)) = f^{(i)}(\boldsymbol{q}(T))$.

An MC2FG $(N, \boldsymbol{q}, (f^{(i)})_{i \in N})$ is said to be a *single-criterion coalition formation game* if forming a coalition derives a single non-transferable value (e.g., the expected number of citations for authors working together on a paper), and all players rank coalitions simply according to these values, irrespective of their scalarisation functions. We use the notation (N, q) to denote a single-criterion coalition formation game.

We say that $f : \mathbb{R}^m \to \mathbb{R}$ is a *linear scalarisation function* if there exists a non-negative vector $\boldsymbol{w} \in \mathbb{R}^m_+$ such that $\sum_{k=1}^m w_k = 1$ and $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} = \sum_{k=1}^m w_k x_k$ for any $\boldsymbol{x} \in \mathbb{R}^m$. We use the notation $(N, \boldsymbol{q}, (\boldsymbol{w}^{(i)})_{i \in N})$ to denote an MC2FG with scalarisation vectors $(\boldsymbol{w}^{(i)})_{i \in N}$. In this paper, we focus on linear scalarisations as a model for user utility.

Solution Concepts An *outcome* of an MC2FG is a partition π of players into disjoint coalitions. Given a partition π of N and a player $i \in N$, let $\pi(i)$ denote the coalition in π containing i. We adapt stability concepts in hedonic games [4, 5] as follows. A partition π of N is said to be *individually rational* if no player strictly prefers staying alone to their own coalitions, i.e., each player $i \in N$ weakly prefers $\pi(i)$ to $\{i\}$. A coalition $S \subseteq N$ *blocks* a partition π of N if every player in $i \in S$ strictly prefers S to $\pi(i)$.

Definition 2. *A partition π of N is said to be core stable (CR) if no coalition $S \subseteq N$ where $S \neq \emptyset$ blocks π.*

We now introduce stability concepts that are immune to deviations by individual players. Consider a player $i \in N$ and a pair of coalitions $S \in 2^N \setminus \mathcal{N}_i$ (possibly the empty set) and $T \in \mathcal{N}_i$. A player i *wants to deviate* from T to S if i prefers $S \cup \{i\}$ to T. A player $j \in S$ *accepts* a deviation of i to S if j weakly prefers $S \cup \{i\}$ to S.

Definition 3. *A deviation of i from T to S is*

- *an NS-deviation if i wants to deviate from T to S.*
- *an IS-deviation if it is an NS-deviation and all players in S accept it.*

Definition 4. *A partition π of N is called* Nash stable (NS) *(respectively,* individually stable (IS)*) if no player $i \in N$ has an NS-deviation (respectively, an IS-deviation) from $\pi(i)$ to another coalition $S \in \pi$ or to $\emptyset$.*

Intuitively, the easier players can deviate, the more stringent the corresponding solution concept is. It is thus easy to see that any Nash stable partition is individually stable. Similarly to the standard hedonic games [4, 5], stable partitions may not necessarily exist as can be seen in the following example.

Example 1. Consider three researchers who can potentially form a research team and write a paper together. They have access to the quality of each group with respect to the three criteria: productivity, creativity, and timeliness. Any pair of researchers can produce some positive correlations, whereas the other coalitions produce nothing or some negative correlations. One can formulate this scenario as an MC2FG $(N, q, (\boldsymbol{w}^{(i)})_{i \in N})$ as follows. The player set is $N = \{1, 2, 3\}$, and $\boldsymbol{q} : 2^N \to \mathbb{R}^3$ is given by

$$\begin{aligned}
&\boldsymbol{q}(\{1\}) = \boldsymbol{q}(\{2\}) = \boldsymbol{q}(\{3\}) = (0, 0, 0)^\top, \\
&\boldsymbol{q}(\{1, 2\}) = (2, 1, 1)^\top, \boldsymbol{q}(\{2, 3\}) = (1, 2, 1)^\top, \boldsymbol{q}(\{1, 3\}) = (1, 1, 2)^\top, \\
&\boldsymbol{q}(\{1, 2, 3\}) = (-1, -1, -1)^\top.
\end{aligned}$$

Player 1 (respectively, 2 and 3) finds the first (respectively, the second and the third) quality measure very important, so the scalarisation functions are given by $\boldsymbol{w}^{(1)} = (1, 0, 0)^\top$ $\boldsymbol{w}^{(2)} = (0, 1, 0)^\top$, and $\boldsymbol{w}^{(3)} = (0, 0, 1)^\top$. The resulting preference profile is as follows:

$$\begin{aligned}
1 &: \{1, 2\} \succ_{\boldsymbol{w}^{(1)}} \{1, 3\} \succ_{\boldsymbol{w}^{(1)}} \{1\} \succ_{\boldsymbol{w}^{(1)}} \{1, 2, 3\}, \\
2 &: \{2, 3\} \succ_{\boldsymbol{w}^{(2)}} \{1, 2\} \succ_{\boldsymbol{w}^{(2)}} \{2\} \succ_{\boldsymbol{w}^{(2)}} \{1, 2, 3\}, \\
3 &: \{1, 3\} \succ_{\boldsymbol{w}^{(3)}} \{2, 3\} \succ_{\boldsymbol{w}^{(3)}} \{3\} \succ_{\boldsymbol{w}^{(3)}} \{1, 2, 3\}.
\end{aligned}$$

This game admits four individually rational partitions: three partitions that consists of a singleton and a pair of the others, $\pi_1 = \{\{1\}, \{2, 3\}\}$, $\pi_2 = \{\{2\}, \{1, 3\}\}$, $\pi_3 = \{\{1\}, \{2, 3\}\}$, and the partition of singletons $\pi_4 = \{\{1\}, \{2\}, \{3\}\}$. It is not difficult to see that none of them is a core stable partition or an individually stable partition.

3 Existence of stable outcomes

As we have seen in the previous section, the set of stable outcomes can be empty in general. Nonetheless, it turns out that for single-criterion coalition formation games core

stability and individual stability can be simultaneously achieved: one can find such an outcome by detecting a sequence of undominated coalitions. A similar construction can be found in [15] for dichotomous hedonic games.

Theorem 1. *Every single-criterion coalition formation game admits a partition that is both core and individually stable.*

Proof. We iteratively find a maximal coalition $S \subseteq N$ of the highest quality $\boldsymbol{q}(S)$, and add S to π. Then, the resulting partition π is both core and individually stable. Observe first that π is core stable, since if there exists a blocking coalition $T \subseteq N$, T would be added to π before any $S \in \pi$ such that $S \cap T \neq \emptyset$. Second, π is individually stable. Notice that no player wants to deviate to a later formed coalition. Moreover, if there exists a player who can IS-deviate to a former formed coalition, this would contradict the maximality of the S.

Moreover, in single-criterion cases, any dynamics under individual stability always converges. Specifically, we define *IS dynamics* to be a procedure by which while the current partition π is not individually stable, we choose an arbitrary player i and a coalition $S \in \pi \cup \{\emptyset\}$ such that i has an IS-deviation to S, and move to the partition $\pi' = (\pi \setminus \{\pi(i), S\}) \cup \{S \cup \{i\}, \pi(i) \setminus \{i\}\}$.

Theorem 2. *In a single-criterion coalition formation game, from an arbitrary initial partition, IS dynamics converges to an individually stable partition.*

Proof. We prove this by an induction on the number of players $|N|$. When $|N| = 1$, our claim clearly holds. Assume that for any $|N| \leq k-1$, IS dynamics converges, and consider the case when $|N| = k$. We will construct a digraph $D = (V, A)$ where V is given by the set of partitions of N, and $(\pi, \pi') \in A$ if and only if π' is *reachable* from π, i.e., there is a player i and a coalition $S \in \pi$ such that i has an IS-deviation to S and $\pi' = \pi \setminus \{S, \pi(i)\} \cup \{S \cup \{i\}, \pi(i) \setminus \{i\}\}$. Suppose towards a contradiction that there is a directed cycle $\mathcal{C} = \{\pi_1, \pi_2, \ldots, \pi_s\}$ in D where $(\pi_t, \pi_{t+1}) \in A$ for all $t \in [s]$; here, we let $\pi_{s+1} = \pi_1$. Now let $S_t \in \operatorname{argmax}_{S \in \pi_t} q(S)$, $q_t = q(S_t)$ for $t \in [s]$, and $q_{s+1} = q_1$. We start by proving the following lemma.

Lemma 1. $q_t \leq q_{t+1}$ *for all* $t \in [s]$.

Proof. Take any $t \in [s]$. Consider the transition from π_t to π_{t+1}. First, if no player moves from or to the coalition S_t, then it is clear that $q_t \leq q_{t+1}$. Second, if there is a player $i \in S_t$ who deviates from S_t to $T \in \pi \cup \{\emptyset\}$, then $q_t < q(T \cup \{i\}) \leq q_{t+1}$. Third, if there is a player $i \in N \setminus S_t$ who deviates from $\pi_t(i)$ to S_t, then it must be the case that $q(S_t) \leq q(S_t \cup \{i\})$ in order for i to be accepted by the players in S_t, and hence $q_t \leq q_{t+1}$. In all cases, we have that $q_t \leq q_{t+1}$.

By Lemma 1, we have that $q_1 \leq \ldots \leq q_s \leq q_1$, implying that the quality of the best coalition does not change along the cycle, i.e., $q_t = q_{t+1}$ for all $t \in [s]$. Now let us focus on the coalition S_1. If there is a player who deviates from S_1 at some point, then the coalition to which the player deviates would produce a higher value than S_1 and hence $q_1 < q_t$ for some $t \in [s]$, a contradiction. Thus, no player moves from S_1 along the cycle $\mathcal{C}$. If there is a player who deviates to S_1 at some point in the cycle, then

the size of the coalition S_1 strictly increases, and hence the dynamics cannot come back to π_1, a contradiction. Therefore, the coalition S_1 remains the same in all the partitions in $\mathcal{C}$, namely, $S_1 \in \pi_t$ for all $t \in [s]$. Now, we again construct a directed graph $D' = (V', A')$ where V' is the set of partitions of $N \setminus S_1$ and $(\pi, \pi') \in A'$ if and only if π' is reachable from π. By the induction hypothesis, D' must be acyclic. However, $\mathcal{C}' = \{\, \pi_t \setminus \{S_1\} \mid t \in [s] \,\}$ forms a directed cycle in D' by the facts that $\mathcal{C}$ forms a directed cycle and that S_1 does not change along $\mathcal{C}$. Hence, we obtain a contradiction, and conclude that from any initial partition the IS dynamics converges.

Due to Theorem 1 and 2, if all players have the same scalarisation function, there always exists a partition that is core and individually stable; moreover, IS dynamics always converges to individual stability.

Corollary 1. *Every multi-criteria coalition formation game admits a core and individually stable partition if all the scalarisation functions are the same. Moreover, IS dynamics always converges to an individually stable partition.*

Proof. Given the scalarisation function $f : \mathbb{R}^m \to \mathbb{R}$, define a single-criterion coalition formation game (N, q') where $q'(S) = f(\boldsymbol{q}(S))$ for each $S \subseteq 2^N$. It is not difficult to see that core or individually stable partitions of the resulting game are also stable partitions of the original game.

We note that Nash stable outcomes may not exist even in single-criterion cases. Consider for instance the two-player game (N, q) where $q(\{1\}) > q(\{1, 2\}) > q(\{2\})$; if player 1 is alone, then player 2 would deviate to his coalition, which would again cause the deviation by player 1. Now, it is natural to wonder what can be said if we have "similar" scalarisation functions. Even in such cases, however, there always exists an MC2FG whose stable partitions are empty.

Theorem 3. *For any positive integer n and for any $0 < \varepsilon < \frac{1}{2}$, there exists an MC2FG $(N, \boldsymbol{q}, \{\boldsymbol{w}^{(i)}\}_{i \in N})$ which admits neither a core nor individually stable partition, where the number of players $|N| = n$, the number of criteria $m = 2$, and $|w_k^{(i)} - w_k^{(j)}| \leq \varepsilon$ for any $i, j \in N$ and any $k \in [m]$.*

Proof. Take any $0 < \varepsilon < \frac{1}{2}$. We choose ε' such that $0 < \varepsilon' < \varepsilon$. Let $c = \frac{1+\varepsilon-\varepsilon'}{2+\varepsilon-\varepsilon'}$. Observe that $\min\{c, 1-\varepsilon\} > \frac{1}{2}$ and hence there exists $\alpha \in \mathbb{R}$ such that $\frac{1}{2} < \alpha < \min\{c, 1-\varepsilon\}$.

Now we construct a two-criteria coalition formation game where $N = [n]$,

$$
\begin{aligned}
&w_1^{(1)} = \alpha + \varepsilon, w_2^{(1)} = 1 - \alpha - \varepsilon, \\
&w_1^{(i)} = w_1^{(j)} = \alpha, \text{ and } w_2^{(i)} = w_2^{(i)} = 1 - \alpha, \text{ for all } i \in N \setminus \{i\},
\end{aligned}
$$

and $\boldsymbol{q} : 2^N \to \mathbb{R}^2$ is given as follows:

$$
\begin{aligned}
&\boldsymbol{q}(\{i\}) = (0, 0) \text{ for all } i \in N, \\
&\boldsymbol{q}(\{1, 2\}) = (1, 0), \boldsymbol{q}(\{2, 3\}) = (1, \varepsilon'), \boldsymbol{q}(\{3, 1\}) = (0, 1 + \varepsilon), \\
&\boldsymbol{q}(\{1, 2, 3\}) = (-1, -1) \\
&\boldsymbol{q}(S) = (-1, -1) \text{ for all } S \not\subseteq \{1, 2, 3\} : |S| \neq 1.
\end{aligned}
$$

Clearly, all players except for $1, 2, 3$ strictly prefer being alone to being together with somebody; hence, these players stay alone at any individually rational partition. The players $1, 2, 3$ strictly prefer pairs to their singletons, and strictly prefer the singletons to the coalition $\{1, 2, 3\}$ and any coalition $S \not\subseteq \{1, 2, 3\}$. Thus, for any individually rational partition π and for all $i = 1, 2, 3$, we have $\pi(i) \subsetneq \{1, 2, 3\}$. Also, it is not difficult to see that $\{2, 3\} \succ_{\boldsymbol{w}^{(2)}} \{1, 2\}$ as the vector $\boldsymbol{q}(\{2, 3\}) = (1, \varepsilon')$ Pareto-dominates $\boldsymbol{q}(\{1, 2\}) = (1, 0)$. Further, we have that $\{3, 1\} \succ_{\boldsymbol{w}^{(3)}} \{2, 3\}$ and $\{1, 2\} \succ_{\boldsymbol{w}^{(1)}} \{3, 1\}$, since

$$\begin{aligned} &w_1^{(3)} q_1(\{1,3\}) + w_2^{(3)} q_2(\{1,3\}) - w_1^{(3)} q_1(\{2,3\}) - w_2^{(3)} q_2(\{2,3\}) \\ &= (1+\varepsilon-\varepsilon') - \alpha \cdot (2+\varepsilon-\varepsilon') > (1+\varepsilon-\varepsilon') - c \cdot (2+\varepsilon-\varepsilon') = 0, \end{aligned}$$

and

$$\begin{aligned} &w_1^{(1)} q_1(\{1,2\}) + w_2^{(1)} q_2(\{1,2\}) - w_1^{(1)} q_1(\{3,1\}) - w_2^{(1)} q_2(\{3,1\}) \\ &= \alpha \cdot (2+\varepsilon) + (\varepsilon^2 + \varepsilon - 1) > \frac{1}{2} \cdot (2+\varepsilon) + (\varepsilon^2 + \varepsilon - 1) > 0. \end{aligned}$$

The resulting preferences restricted to $\{1, 2, 3\}$ are the same as in Example 1, meaning that the instance has neither a core nor an individually stable partition.

Another implication of Theorem 3 is that even if the number of criteria is much smaller than the number of players there exists a two-criteria MC2FG which does not admit a stable partition.

4 Algorithms

Because none of stable partitions necessarily exists in an MC2FG, there is no algorithm that can guarantee a stable partition as an outcome. However, because we know (Theorem 1 and Corollary 1) that if there is only one criterion or if all the agents have the same scalarisation function, individually stable partitions do exist, we *expect* the chances of stable partitions existing in a random MC2FG to increase as the number of criteria decreases. In order to test this hypothesis, we devise heuristic algorithms for constructing stable partitions. Here, we do not focus on the core since checking core stability is computationally intractable (see e.g. [19]): we need to iterate through all subsets of players to see whether it is a blocking coalition.

We aim for our algorithms to minimise the number of questions that need to be asked to each agent. This is essential, as asking such questions to people can be time-consuming — both in terms of time required by the humans, and the time the system needs to wait until an answer is received — and experienced as hindrance by these humans.

We define a so-called *local search (LS)*, or *best-response*, algorithm for MC2FGs called *local stability search (LSS)*. LSS starts from a partition, π. At each time-step, the algorithm selects an agent, i, computes whether there exists a deviation (Definition 3) from $\pi(i)$ to any other coalition $T \in \pi \setminus \pi(i)$, and if it does performs the deviation. When there are no more deviations for any agent, the partition is stable.

A key aspect of LSS is that at any given iteration, LSS may not be able to decide whether an agent prefers an alternative coalition over another before explicitly asking that

Algorithm 1: `LSS`$(N, \boldsymbol{q}, \pi,$ `checkDeviation`$)$

```
1  C ← a set of simplex constraints on w^(i), C^(i), for each i ∈ N
2  stable ← false
3  while ¬stable ∨ ¬timeout() do
4      stable ← true
5      foreach i ∈ N do
6          foreach T ∈ (π \ π(i)) that i could join do
7              if (T ∪ {i} ≻_P T) ∧ (T ∪ {i} ≻_P π(i)) then
8                  π ← (π \ {π(i), T}) ∪ {π(i) \ {i}, T ∪ {i}}
9                  stable ← false
10                 continue from top while-loop (line 3)
11             end
12         end
13     end
14     foreach i ∈ N do
15         // for all agents (in random order), check for and perform deviations:
16         foreach T ∈ (π \ π(i)) that i could join do
17             possibleDeviation ← checkDeviation(i, π(i), T, C)
18             if possibleDeviation then
19                 π ← (π \ {π(i), T}) ∪ {π(i) \ {i}, T ∪ {i}}
20                 stable ← false
21                 continue from top while-loop (line 3)
22             end
23         end
24     end
25 end
26 if stable then return π ;
27 else return No stable partitioning was found ;
```

agent. For example, imagine that LSS is currently considering a partition π, and knows nothing about the $\boldsymbol{w}^{(i)}$ of an agent, i. When considering whether i wants to deviate from $\pi(i)$ to say, a coalition $T \in \pi$, we must know whether, $\boldsymbol{w}^{(i)} \cdot \boldsymbol{q}(\pi(i)) < \boldsymbol{w}^{(i)} \cdot \boldsymbol{q}(T \cup \{i\})$, where "$\cdot$" denotes the inner product. When for example, $\boldsymbol{q}(\pi(i)) = (0, 3)$ and $\boldsymbol{q}(T \cup \{i\}) = (1, 4)$, i will always prefer to deviate, as there is no $\boldsymbol{w}^{(i)}$ for which $\boldsymbol{w}^{(i)} \cdot \boldsymbol{q}(\pi(i)) \geq \boldsymbol{w}^{(i)} \cdot \boldsymbol{q}(T \cup \{i\})$. However, if $\boldsymbol{q}(\pi(i)) = (2, 3)$ and $\boldsymbol{q}(T \cup \{i\}) = (1, 4)$, there are possible values for $\boldsymbol{w}^{(i)}$ that would make i prefer $\pi(i)$. In such cases we have to *elicit* the preferences of agent i with respect to these two vectors.

LSS is provided in Algorithm 1 and is parameterised by the agents and quality function of an MC2FG, i.e., N and $\mathbf{q}$. However, we assume that we have no *direct* access to $f^{(i)}$ (i.e., $\boldsymbol{w}^{(i)}$). In fact, LSS only knows that each $\boldsymbol{w}^{(i)}$ adheres to the simplex constraints, i.e., $\sum_x w_x^{(i)} = 1$ and $\forall x : 0 \leq w_x^{(i)} \leq 1$. Therefore, it creates a set of sets of constraints on line 1 containing the simplex constraints for each $\boldsymbol{w}^{(i)}$.

LSS is also parameterised by a starting coalition π, which can e.g., be initialised randomly. Finally LSS is parameterised by a function `checkDeviation`. This function checks whether a deviation exists, and thus requires different implementations for NS-

Algorithm 2: `checkNSDeviation`$(i, \pi(i), T, \mathcal{C}^{(i)})$

```
1  maxDiffNew ← max_{w^(i)} w^(i) · (q(T ∪ {i}) − q(π(i))) s.t. C^(i)
2  maxDiffOld ← max_{w^(i)} w^(i) · (q(π(i)) − q(T ∪ {i})) s.t. C^(i)
3  if maxDiffOld ≥ 0 ∧ maxDiffNew > 0 then
4      // not enough information, ask agent i:
       prefNew ← askAgent(T ∪ {i} ≻_{f^(i)} π(i))
5      if prefNew then
6          C^(i) ← C^(i) ∪ {w^(i) · (q(T ∪ {i}) − q(π(i))) > 0}
7          return true
8      else
9          C^(i) ← C^(i) ∪ {w^(i) · (q(T ∪ {i}) − q(π(i))) ≤ 0}
10         return false
11     end
12 else return maxDiffNew > maxDiffOld ;
```

deviations (Algorithm 2), and IS-deviations (Algorithm 3). It is also this function that will elicit preferences from the agents.

In the main loop (line 3-25), LSS iterates over all agents two times, and checks whether it has a deviation it wants to perform (lines 7 and 17). The first time LSS loops over all agents, it checks whether i can deviate to a coalition T for which $\boldsymbol{q}(T \cup \{i\})$ Pareto-dominates, i.e., is better or equal in all criteria and better in at least one criterion than, both $\boldsymbol{q}(T)$ and $\boldsymbol{q}(\pi(i))$; if that is the case, both i and the agents in T will prefer that definition, will thus allow both an NS- and IS-deviation. If such a deviation exists, it is performed (line 8). The second time LSS loops over the agents, it checks for each agent i whether there is an NS-deviation or an IS-deviation using an NS- or IS-specific subroutine. If such a deviation exists, the deviation is performed (line 19). When none of the agents have a deviation LSS terminates. In order to check whether an agent has a deviation, we need a specific algorithm for each type of deviation. For NS-deviations, the algorithm is given in Algorithm 2. The algorithm is called with an agent i that may want to deviate from $\pi(i)$ to T, given the known constraints on $\boldsymbol{w}^{(i)}$, $\mathcal{C}^{(i)}$. On the first two lines, *linear programs (LPs)* are run to calculate the maximal possible difference in utility $\boldsymbol{w}^{(i)} \cdot \boldsymbol{q}(S)$ if the new coalition, i.e., $S = T \cup \{i\}$, is preferred over the old coalition, i.e., $\pi(i)$, resp. if the old coalition is preferred over the new one, given the known constraints $\mathcal{C}^{(i)}$. When both these values are positive, it is both possible that the old coalition has a higher utility for i and that the new coalition has a higher utility for i. In other words, LSS cannot determine which coalition is preferred by i and it must thus ask the agent directly (line 3). We denote this asking the agent as `askAgent`$(T \cup \{i\} \succ_{f^{(i)}} \pi(i))$. We assume the agent will always answer truthfully with either `true` or `false`. When an agent answers a comparison question, for example, it states that `askAgent`$((2,3) \succ_{f^{(i)}} (1,4)) \rightarrow$ `true`, this imposes a constraint on $\boldsymbol{w}^{(i)}$. In this case, it imposes the constraint $\boldsymbol{w}^{(i)} \cdot (2-1, 3-4) > 0$. In general, the imposed constraints are:

$$\boldsymbol{q}(S) \succ_{f^{(i)}} \boldsymbol{q}(T) \implies \boldsymbol{w}^{(i)} \cdot (\boldsymbol{q}(S) - \boldsymbol{q}(T)) > 0, and$$

Algorithm 3: `checkISDeviation`$(i, \pi(i), T, \mathcal{C})$

```
1  nsDeviation ← checkNSDeviation(i, π(i), T, C^(i))
2  if nsDeviation then
3      foreach j ∈ T do
4          maxDiffNew ← max_{w^(j)} w^(j)·(q(T∪{i})−q(T)) s.t. C^(j)
5          maxDiffOld ← max_{w^(j)} w^(j)·(q(T)−q(T∪{i})) s.t. C^(j)
6          if maxDiffOld > 0 ∧ maxDiffNew ≥ 0 then
7              //not enough information, ask agent j:
8              prefOld ← askAgent(T ≻_{f^(j)} T ∪ {i})
9              if prefOld then
10                 C^(j) ← C^(j) ∪ {w^(j) · (q(T) − q(T ∪ {i})) > 0}
11                 return false
12             else
13                 C^(j) ← C^(j) ∪ {w^(j) · (q(T) − q(T ∪ {i})) ≤ 0}
14             end
15         else if maxDiffOld > 0 then
16             return false
17         end
18     end
19     return true
20 else return false ;
```

$$\neg(\boldsymbol{q}(S) \succ_{f^{(i)}} \boldsymbol{q}(T)) \implies \boldsymbol{w}^{(i)} \cdot (\boldsymbol{q}(S) - \boldsymbol{q}(T)) \leq 0.$$

Therefore, by eliciting such constraints through asking agents for comparisons between quality vectors, LSS can learn the relevant preference information of the agents. LSS adds the constraints to $\mathcal{C}^{(i)}$ (line 6 and 9).

When LSS is run with `checkNSDeviation` (Algorithm 2) as `checkDeviation`, it will try to find a Nash-stable partitioning. However, we may want to consider weaker stability concepts; for individual stability, we need to use `checkISDeviation` (Algorithm 3) instead. Because individual stability imposes extra constraints on deviations over Nash-stability, `checkISDeviation` first calls `checkNSDeviation` (line 1), to check whether i wants to deviate. Then, if that is indeed the case, it loops over all the agents of the coalition i wants to deviate to, T, to check whether none of these agents lose utility by i joining T (line 3–18). This is done using similar LPs as for the agent that wants to deviate (line 4 and 5). Again, the algorithm might not be able to determine this from the current constraints, and thus elicits comparisons from the appropriate agent. Note that when one agent in T loses utility, the `checkISDeviation` terminates immediately (line 11). This is because we want to minimise the number of questions asked.

We note that on line 6 and 16 of Algorithm 1 (LSS), we may not allow i to deviate to arbitrary $T \in \pi \setminus \pi(i)$. For example, we may impose social network constraints [11], i.e., that the agents are embedded in a graph representing which agents know each other, and only allow coalitions that are connected subgraphs.

5 Experiments

In this section we empirically test the LSS algorithm for both Nash stability and individual stability. We initialise the partition at the start of each run as the *partition of singletons*, i.e., initially each agent is in a separate coalition. As a baseline, we compare to an algorithm that does not employ linear programs, but only tests for Pareto-dominance instead, i.e., Algorithm 1 is the same, but in the `checkDeviation` subroutines (Algorithms 2 and 3) the linear programming steps (e.g., line 1–2 in Algorithm 2) are skipped, and instead the agent is always asked for a comparison. We refer to this baseline algorithm as *always ask stability search (AASS)*.

We implemented all algorithms in Python 3, making use of (the default solver of) the PuLP library (version 1.6.1) for linear programs. We ran the experiments on a MacBook Pro, with a 2.9 GHz Intel Core i5 processor and 16GB memory, running macOS Sierra (version 10.12.1).

5.1 Test problems

In order to test the performance of LSS and AASS we make use of two test classes of MC2FGs. A `Random` instance, is one in which every possible coalition $S \subseteq N$ has a randomly drawn quality vector $\boldsymbol{q}(S)$, from a uniform distribution on the unit hypercube of m dimensions (i.e., between the origin, $\mathbf{0}$, and the vector containing only ones, $\mathbf{1}$). The scalarisation function for each agent, i, is linear, with a weight vector $\boldsymbol{w}^{(i)}$ that is drawn independently from a uniform distribution on the weight simplex.

The second problem, `Author` $\times$ `Author`, is inspired on the example of scientists writing papers together. Imagine we have n authors (with scalarisation functions generated in the same way as for the `Random` instances) whom will be advised to work together in coalitions by a recommender system. This system interacts with the agents by asking them whether they would prefer to be in one of two proposed coalitions. For each coalition, the recommender system presents the expected quality $\boldsymbol{q}$ in m dimensions (e.g., expected impact and expected novelty when writing a paper together), of the coalitions to the agents. It is therefore essential that the number of questions posed to the agents is minimised, as interaction with human decision makers is the slowest part of the process. The quality vectors are formed by summing over randomly drawn agent quality vectors $\mathbf{v}_i$ for each agent (containing values between 1 and 4 drawn independently from a uniform distribution), and subtracting a group size penalty:

$$\boldsymbol{q}(S) = (-2^{|S|-1})\mathbf{1} + \sum_{i \in S} \mathbf{v}_i.$$

Because of the group size penalty, stable partitions will typically consist of coalitions of 3 or 4 agents. Furthermore, because of this structure, we have empirically found that Nash stable partitions typically do exist in `Author` $\times$ `Author` instances.

5.2 Random

To test whether we can find stable partitions with LSS, we first run LSS on Random MC2FG instances, as defined above. Note that we do not test AASS separately, as they have the same number of iterations; the only difference between the two is when they ask the agents to compare quality vectors.

In Figure 1, proportion of instances (of 50 in total) for which LSS found a Nash or individually stable partition within 1000 iterations for varying numbers of agents (top left) and criteria (top right).

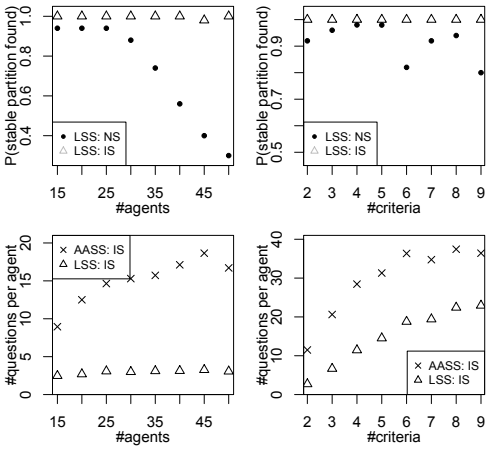

Fig. 1. The proportion of instances for which a stable partition was found by LSS using the Nash stability criterion and Individual stability criterion within 1000 iterations (top) and average number of questions asked per agent until an IS-stable partitioning was found (bottom), as a function of (left) the number of agents in 50 2-criteria Random instances, and (right) the number of criteria in 50 20-agent Random instances.

We observe that for the individual stability criterion, stable partitions were nearly always found (in only 2 out of the 800 Random instances an individually stable coalition was not found by LSS), while for Nash stability this is not the case. This is according to expectation, as it is more difficult to reach a Nash stable partition (as Nash stability is a stronger stability concept than individual stability). Furthermore, we observe that the proportion of stable partitions found goes down as a function of the number of agents (as expected), but does not change significantly as a function of the number of criteria. This is surprising as more criteria make the likelihood of agents disagreeing more likely. We thus conclude that MC2FGs become significantly harder as the number of agents in the problem increases.

Secondly, we measure how many questions we need to ask per agent in order to find stable partitions. This is important, as this may correspond to asking humans for their preferences, which can be time-consuming and experienced as hindrance by these humans. Because we have not always found Nash stable partitions, we focus on individual stability only for the Random MC2FG instances. From Figure 1 (bottom left) we observe that for LSS leads to significantly less questions asked per agents than AASS across all numbers of agents. The highest number of questions per agents (at 40 agents) was 3.26 for LSS and 18.65 for AASS. LSS scales better in the number of agents than AASS; when we fit a line for the number of questions per agent as a function of the number of agents, we obtain a slope of 0.017 questions per agent per additional agent for LSS, and a slope of 0.22 for AASS.

LSS needs less questions per agent than AASS across different numbers of criteria (Figure 1; bottom right). However, the difference in slope, i.e., the number of questions per agent per additional criterion is not significant (2.96 for LSS, versus 3.36 for AASS).

We conclude that LSS can be used to find stable partitions in Random MC2FG instances, but that Nash stable partitions are harder to find than individually stable

partitions. Furthermore, LSS can significantly decrease the number of comparisons that need to be made by the agents when looking for individually stable partitions, and LSS scales better in the number of agents than a naive question asking scheme like AASS.

5.3 Author × Author

In order to test the performance of LSS with respect to AASS, we test the algorithms on our real-world inspired Author × Author problem. Important features of this problem is that stable coalitions typically consist of 3- or 4-agent coalitions, and that stable partitions typically exist. Indeed, we did not come across any instance in our experiments for which a stable partition was not found. This latter feature provides us with the opportunity to study the number of questions asked *until* a stable partition is reached, without having to worry about *whether* a stable partition exists.

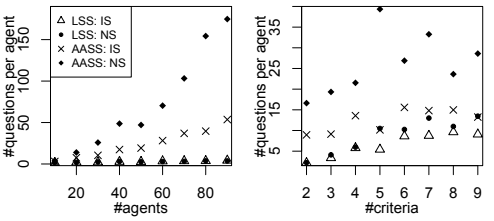

Fig. 2. The average number of questions asked per agent until an individual or Nash stable partitioning was found as a function of (left) the number of agents in 2-criteria Author × Author instances, and (right) the number of criteria in 20-agent Author × Author instances.

We compare LSS to AASS, and finding individually stable partitions to finding Nash stable partitions on Author × Author instances. For all Author × Author instances Nash and individually stable partitions were found within 1000 iterations of the main loop of LSS/AASS (Algorithm 1, lines 3–25). While only singleton initialisation is displayed in Figure 2, we also compared to undominated initialisation, but like for Random instances found no significant difference.

When comparing finding Nash stable partitions to individually stable partitions for varying numbers of agents (Figure 2 (left)), we observe that AASS performs significantly worse for individual stability than for Nash stability in terms of the number of questions, there is no significant difference between the number of questions until a Nash stable partition and an individually stable partition is found by LSS. This is a surprising result, as it is typically much harder to find Nash stable partitions (as can be seen from the AASS curve for the number of questions asked), and it does take more iterations to find a Nash stable partition than an individually stable partition (on average 253 versus 144 iterations for 90 agent instances). Furthermore, both for individual stability and for Nash stability, LSS scales better than AASS in terms of the number of questions per agent.

For 80-agent 2-criteria Author × Author instances, LSS required 3.7 questions on average per agent. When we consider the example use case of matching small groups of authors for an event, asking authors to compare 3 or 4 groups is probably feasible. On the other hand, if we employ a more naive question-asking scheme, i.e., AASS, the numbers are 40 for individual stability, and 154, which probably would not be feasible. We thus conclude that LSS can keep the number of questions that need to be asked to agents can be kept at feasible numbers even for higher number of agents.

When we compare Author × Author instances of 20 agents for varying numbers of criteria (Figure 2 (right)), LSS also outperforms AASS by a large margin for both Nash stability and individual stability. For higher criteria, LSS using individual stability is slightly more efficient than LSS using Nash stability. For 9 criteria, 20 instances required

10.6 questions per agents for finding an individually stable partition with LSS, and 14.4 for finding Nash stable partitions. We conclude that for this real-world inspired problem, LSS can reduce the number of questions that need to be asked to agents can be kept at reasonable numbers, even for higher numbers of criteria.

6 Discussion

In this paper, we proposed the *multi-criteria coalition formation game (MC2FG)* for forming stable partitions while having limited access to the preferences over different possible coalitions for each agent. This is important as agents may either not want to divulge their complete preference profiles for social or privacy reasons, but more importantly might not even be able to specify their utilities a priori to begin with. Instead, MC2FGs model the quality of coalitions as vectors containing different agent-independent metrics corresponding to different criteria that agents may have. However, the agents may have different preferences with respect to these criteria. To model this, in MC2FGs each agent, i, has a private utility function, $f^{(i)}$, that takes the multi-criteria quality vector of a coalition, and produces a scalar utility. Because these functions are private, we cannot use these to check whether a proposed coalition is stable. However, some a priori assumptions about $f^{(i)}$ can be made, and we can increase our knowledge about $f^{(i)}$, by asking agent i to compare two coalitions. In this paper we made the a priori assumption that each $f^{(i)}$ is a linear function.

Because (Nash and individually) stable partitions need not exist in MC2FGs, we proposed a local search algorithm that we call *local stability search (LSS)* to find stable partitions where possible. We showed empirically that LSS is able to discover stable partitions. By exploiting the additional knowledge gained by asking agents for comparisons between quality vectors for different coalitions (via LPs), we show empirically that the number of comparisons that need to be asked from agents can be kept to a minimum.

Because stable partitions need not exist in general, in future research we aim to find subclasses of MC2FGs in stable partitions are guaranteed to exist. Specifically, we aim to find realistic subclasses corresponding to use cases like the `Author` $\times$ `Author` problem. Furthermore, we aim to improve the `Author` $\times$ `Author` problem, using studies on scientific cooperation [1, 12] to redefine the utility functions.

More generally, we have applied a utility-based approach to multi-criteria multi-agent decision making. In our model, all agents in a coalition share a value vector, but may have different private utilities w.r.t. this value vector, as their preferences with respect to the criteria may vary. We believe that this kind of model is realistic for many real-world decision-making problems in which there are heterogeneous agents, e.g., colleagues in different phases of their careers cooperating in a project. We aim to investigate how this perspective can be applied to other multi-agent decision-making models, such as *(multi-objective) coordination graphs* [9, 17], *(cooperative) Bayesian games* [14] and *(partially observable) stochastic games* [10, 13]. Finally, we want to investigate possibility of noisy pairwise comparisons [18].

Acknowledgements

We are grateful for very useful comments by the anonymous reviewers at ADT and CoopMAS, and thank Edith Elkind for her helpful suggestions. Diederik M. Roijers is a postdoctoral fellow of

the Research Foundation – Flanders (FWO). This work is in part supported by the EC-FP7 under grant agreement no. #611153 (TERESA).

References

1. M. Ackerman and S. Brânzei. The authorship dilemma: alphabetical or contribution? In *AAMAS 2014*, pages 1487–1488, 2014.
2. M. Balcan, A. D. Procaccia, and Y. Zick. Learning cooperative games. In *IJCAI 2015*, pages 475–481, 2015.
3. N. Benabbou and P. Perny. Solving Multi-Agent Knapsack Problems. In *ECAI 2016*, pages 1318–1326, 2016.
4. S. Banerjee, H. Konishi, and T. Sönmez. Core in a simple coalition formation game. *Social Choice and Welfare*, 18(1):135–153, 2001.
5. A. Bogomolnaia and M. O. Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2):201–230, 2002.
6. G. Chalkiadakis, E. Elkind, and M. Wooldridge. Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(6):1–168, 2011.
7. P. Faliszewski, E. Elkind, and M. Wooldridge. Boolean combinations of weighted voting games. In *AAMAS 2009*, pages 185–192, 2009.
8. F.R. Fernández, M.A. Hinojosa, and J. Puerto. Core concepts in vector-valued games. *Journal of Optimization Theory and Applications*, 112(2): 331–360, 2002.
9. C. Guestrin, D. Koller, D., and R. Parr. Multiagent planning with factored MDPs. In *NIPS 2002*, pages 1523–1530, 2002.
10. E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI 2004*, pages 709–715, 2004.
11. A. Igarashi, and E. Elkind. Hedonic games with graph-restricted communication. In *AAMAS 2016*, pages 242–250, 2016.
12. J. Kleinberg, and S. Oren. Mechanisms for (mis) allocating scientific credit. In *STOC 2011*, pages 529–538, 2011.
13. M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *ICML 1994*, pages 157–163, 1994.
14. F. A. Oliehoek, S. Whiteson, and M. T. J. Spaan. Exploiting structure in cooperative Bayesian games. In *UAI 2012*, pages 654–664, 2012.
15. D. Peters. Complexity of Hedonic Games with Dichotomous Preferences. In *AAAI 2016*, pages 579–585, 2016.
16. D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 47:67–113, 2013.
17. D. M. Roijers, S. Whiteson, and F. A. Oliehoek, Computing convex coverage sets for faster multi-objective coordination. *Journal of Artificial Intelligence Research*, 52:399–443, 2015.
18. D. M. Roijers, L. M. Zintgraf, and A. Nowé Interactive Thompson Sampling for Multi-Objective Multi-Armed Bandits. In *ADT 2017*, 2017. (To appear.)
19. S. C. Sung and D. Dimitrov, On core membership testing for hedonic coalition formation games. *Operations Research Letters*, 35(2):155–158, 2007.
20. T. Tanino. Multiobjective cooperative games with restrictions on coalitions. In *Multiobjective Programming and Goal Programming*, pages 167–174. Springer, 2009.