Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Forming *k* coalitions and facilitating relationships in social networks[☆]

Liat Sless[a], Noam Hazon[b,*], Sarit Kraus[a], Michael Wooldridge[c]

[a] *Department of Computer Science, Bar-Ilan University, Israel*
[b] *Department of Computer Science, Ariel University, Israel*
[c] *Department of Computer Science, University of Oxford, UK*

A R T I C L E   I N F O

Article history:
Received 21 May 2016
Received in revised form 25 February 2018
Accepted 13 March 2018
Available online 20 March 2018

Keywords:
Coalition formation
Additively separable hedonic games
Social networks

A B S T R A C T

In this paper we relax two common assumptions that are made when studying coalition formation. The first is that any number of coalitions can be formed; the second is that any possible coalition can be formed. We study a model of coalition formation where the value depends on a social network and exactly *k* coalitions must be formed. Additionally, in this context we present a new problem for an organizer that would like to introduce members of the social network to each other in order to increase the social welfare or to stabilize a coalition structure.
We show that, when the number of coalitions, *k*, is fixed and there are not many negative edges, it is possible to find the coalition structure that maximizes the social welfare in polynomial time. Furthermore, an organizer can efficiently find the optimal set of edges to add to the network, and we experimentally demonstrate the effectiveness of this approach. In addition, we show that in our setting even when *k* is fixed and there are not many negative edges, finding a member of the core is intractable. However, we provide a heuristic for efficiently finding a member of the core that also guarantees a social welfare within a factor of 1/2 of the optimal social welfare. We also show that checking whether a given coalition structure is a member of the core can be done in polynomial time. Finally, we consider the problem faced by an organizer who would like to add edges to the network in order to stabilize a specific coalition structure core: we show that this problem is intractable.

1. Introduction

Coalition formation is one of the fundamental problems in multi-agent systems research [10]. Broadly speaking, coalition formation is concerned with partitioning a population of agents into disjoint teams (or *coalitions*), typically with the aim that some system-wide performance measure is maximized. Most studies of coalition formation in the multi-agent systems community are based on models from the field of cooperative game theory (e.g., characteristic function games [35], non-transferable utility games [17], or hedonic games [18]). While these models have proven to be extremely useful in identifying the key issues in coalition formation, they are typically too abstract to be used in real-world cooperative scenar-

☆ This paper extends an earlier conference paper [36].
* Corresponding author.
E-mail addresses: liat.sless@gmail.com (L. Sless), noamh@ariel.ac.il (N. Hazon), sarit@cs.biu.ac.il (S. Kraus), mjw@cs.ox.ac.uk (M. Wooldridge).

https://doi.org/10.1016/j.artint.2018.03.004
0004-3702/© 2018 Elsevier B.V. All rights reserved.

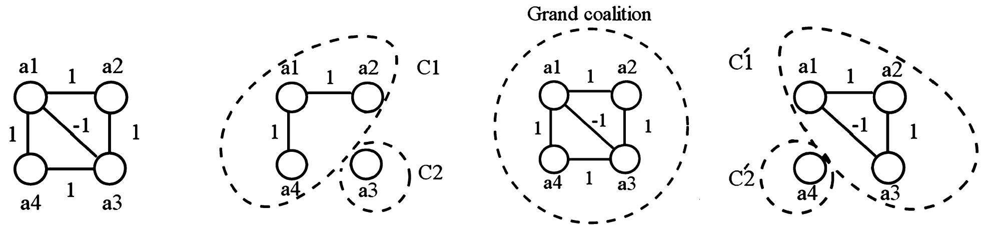
ios. For example, models based on transferable utility games in characteristic function form simply assign a value $v(C)$ to every possible coalition C, indicating the payoff that would be available to coalition C, should they choose to work together. Such a model does not attempt to capture issues such as the possibility that not all coalitions may form, or the complex issues surrounding precisely how the coalition will work together. This limits the applicability of such models and techniques. A challenge for multi-agent systems researchers is therefore to investigate the extent to which traditional models of coalition formation can be enriched in order to adequately model real-world scenarios, while ensuring that such additional expressive power does not bring with it computational intractability.

One gross simplifying assumption made in most cooperative game models is that *any number of coalitions can be formed*. However, in many real-world scenarios there is a strong restriction on the number of possible coalitions. For example, consider a manager facing the challenge of allocating workers to a project that requires the execution of k different tasks. Since there are k tasks to carry out, any coalition structure with less than k coalitions will not complete the project, and any coalition structure with more than k coalitions includes redundant activity. Another common assumption, embodied within many models, is that *any possible coalition can be formed*. In real-world scenarios, there are typically many factors that prohibit the formation of some coalitions. Indeed, an issue that should often be taken into account is the interpersonal relationships between potential coalition members. For example, irrespective of the personal skills of the relevant individuals, in a mission-critical situation, we would typically not seek to form a coalition between individuals that were not acquainted, or that did not have a proven track record of being able to cooperate effectively. It is, of course, precisely these concerns that lead organizations to undertake team-building initiatives and the like, with the goal of either establishing or further strengthening interpersonal relationships between potential team members.

In this paper we address the problem of how cooperative game theory models, concepts, and techniques can be adapted to capture these concerns. The environment is modeled as follows. We use an undirected weighted graph to represent a social network – vertices are agents, edges indicate acquaintance relationships between agents, and the weight of an edge indicates the strength of the relationship. Weights may be positive or negative, and negative values imply that the respective agents are not able to cooperate successfully. The utility function of an agent (being a part of a coalition) is given by the sum of weights of edges in the social network with agents in her coalition with whom she is acquainted. The value of a coalition is defined to be the sum of utilities of agents in that coalition. We further assume that there is a centralized entity (the “organizer” hereafter) whose goal is to create coalitions to carry out k tasks. Thus, the organizer wants to build a coalition structure containing k coalitions. We sometimes assume that the organizer has some capability to *adapt* the game, by *building relationships between agents* (adding edges to the social network), although such relationship building is assumed to have an associated cost. Thus, the problem of the organizer involves trading off the cost of creating such relationships against the benefits that this brings to coalition formation.

Within this setting, we consider two key problems. The first is that of creating coalition structures with exactly k coalitions, which maximize social welfare (i.e., the sum of values of coalitions within a coalition structure). We show that this problem can be solved in polynomial time if there are only a small number of negative edges in the social network and k is fixed. If one of these conditions does not hold, we show that the problem becomes NP-complete. We note that we do not impose any other restrictions on the structure of the social network, in contrast to much related work [7]. We also show that the coalition structure that maximizes the social welfare is not necessarily a member of the core, but it still has some properties of stability. We then consider the problem faced by an organizer, who would like to add edges to the network in order to increase the social welfare. We present a polynomial time algorithm for this purpose and evaluate its performance on some sample real-world networks. Our experimental results demonstrate the benefits of the adaption made by the organizer, and that a naive algorithm will perform far from optimally. To further improve the running time of the optimal algorithm, we also provide an efficient greedy heuristic, and show that its performance is near optimal in practice.

We then consider the computational tractability of *core stable* coalition structures to carry out exactly k tasks. Coalition structures that maximize social welfare are relevant for situations in which organizers can force individuals to be in groups. In situations where this is not possible, we must consider *selfish* coalition formation, and in such settings the relevant questions are whether or not a given coalition structure is *core stable* (i.e., whether or not any subset of agents has any incentive to deviate from a coalition), and whether or not there exists any coalition structure that is core stable [10]. We find that answering both questions is intractable (NP-hard) even when k is fixed, but we provide a polynomial time algorithm for deciding whether or not a given coalition is core stable when there are only a small number of negative edges in the social network. We also provide a heuristic for efficiently finding a member of the core that guarantees a social welfare within a factor of $1/2$ of the optimal social welfare. In order to test the effectiveness of the heuristic, we conducted an experimental evaluation on several real-world networks. Surprisingly, in all the networks we tested, the core always existed, and the coalition structure maximizing the social welfare was always a member of the core (thus a heuristic for finding a member of the core is not needed). However, our heuristic turns out to be extremely useful for finding a member of the strict core (see the definition in next section). Finally, we consider the problem faced by an organizer, who would like to add edges to the network in order to stabilize a specific coalition structure. We show that this problem is closely related to the cost of stability problem, and indeed both problems are NP-hard. We conclude by surveying related work and suggest directions for future work.



(a) The social network

(b) $u(a_1, C_1) = 2$
$u(a_2, C_1) = 1$
$u(a_4, C_1) = 1$
$u(a_3, C_2) = 0$
$SW(P) = 4$

(c) $u(a_1, C_1) = 1$
$u(a_2, C_1) = 2$
$u(a_4, C_1) = 2$
$u(a_3, C_2) = 1$
$SW(P) = 6$

(d) $u(a_1, C_1) = 0$
$u(a_2, C_1) = 2$
$u(a_4, C_2) = 0$
$u(a_3, C_1) = 0$
$SW(P) = 2$

Fig. 1. Agent utilities using different coalition structures.

2. Preliminary definitions

Let $A = \{a_1, \ldots, a_n\}$ be a finite, non-empty set of agents, and let $G = (A, E, \omega)$ be an undirected weighted graph with no self loops, representing the relations between agents. Each edge $(a_i, a_j) \in E$ is associated with a weight $\omega(a_i, a_j) = \omega(a_j, a_i) \in \mathbb{Z}$, representing the relationship strength or the affinity between agents a_i and a_j. If $(a_i, a_j) \notin E$ then we assume that $\omega(a_i, a_j) = 0$. Note that the weight on an edge (a_i, a_j) represents the ability of agents a_i and a_j to cooperate and work together successfully. We refer to G as the *social network*. A coalition $C \subseteq A$ is a subset of agents; we do not require that agents in a coalition form a connected component in the corresponding social network. We let $u(a_i, C)$ denote the utility that agent a_i would obtain from being in coalition C. This value is simply the sum of edge weights corresponding to the immediate neighbors of a_i that are members of C, that is:

$$u(a_i, C) = \sum_{a_j \in C} \omega(a_i, a_j).$$

We denote the value of a coalition $C \subseteq A$ by $V(C)$, and define this value to be the sum of utilities of members of C, that is:

$$V(C) = \sum_{a_i \in C} u(a_i, C).$$

We assume there is a central organizer, who desires to establish coalitions in order to perform $0 < k \leq N$ tasks. Let $\Pi_k(A)$ denote the set of partitions of agents A that contains exactly k non-empty subsets. We refer to elements of $\Pi_k(A)$ as *coalition structures*, and typically use $P, P', \ldots$ to denote such coalition structures. With respect to P, let $\sigma_P : A \to P$ be the function mapping agents to their coalitions, i.e., if $P \in \Pi_k(A)$, $C_i \in P$, and $a \in C_i$ then $\sigma_P(a) = C_i$. If the context of P is clear, we will omit reference to it.

We denote the *social welfare* of the coalition structure $P \in \Pi_k(A)$ by $SW(P)$ (see, e.g., [35]):

$$SW(P) = \sum_{C \in P} V(C).$$

When we need to explicitly identify the edges of the social network E (with the corresponding weight function) in a social welfare computation, we will write $SW(P, E)$. A simple example illustrating the above definitions is presented in Fig. 1. As an aside, note that the model described above can be understood as a special case of *symmetric additively separable hedonic games* [10].

We will address scenarios in which a centralized organizer has the ability to adapt the game by adding edges to the social network G. Specifically, if $(a_i, a_j) \notin E$, then we assume the organizer is able to add an edge connecting a_i to a_j such that subsequently $\omega(a_i, a_j) = 1$ (we will later relax this assumption). Intuitively, adding an edge (a_i, a_j) implies creating a social relationship between agents a_i and a_j where no such relationship existed before. In real-world scenarios, such working relationships are facilitated by a range of mechanisms, such as team-building exercises. We assume that the organizer incurs a fixed cost, denoted by α, for every edge added.

Let E^+ be the set of edges that the organizer is adding to the graph, and E^{new} be the set of edges of G after the addition done by the organizer, with corresponding weights ω^{new}; that is:

$$\omega^{new}(a_i, a_j) = \begin{cases} 1 & (a_i, a_j) \in E^+ \\ \omega(a_i, a_j) & (a_i, a_j) \in E \end{cases}$$

Many solution concepts have been developed for cooperative games, and we now describe how the most prominent of these may be adapted to our model (i.e., where coalition structures contain exactly k coalitions). To better understand our adaption, we first recall the standard definition of a blocking coalition.

Definition 1. Let $P = \{C_1, C_2, \dots, C_k\}$ be a coalition structure. A coalition $B \subseteq A$ is *blocking* if $\forall a \in B : u(a, B) > u(a, \sigma_P(a))$

According to this definition, if a set of agents B forms a blocking coalition, it means that all of them prefer to deviate and form a new separate coalition, causing a deviation from $P = \{C_1, C_2, \dots, C_k\}$ to $P' = \{B, C_1 \setminus B, C_2 \setminus B, \dots, C_k \setminus B\}$. Note that the resulting coalition structure P' may contain any number of coalitions between 1 and $k + 1$. For example, if B is the grand coalition, i.e., $\forall i : C_i \subseteq B$, then $P' = \{B\}$, a coalition structure containing exactly one coalition. On the other hand, if $\nexists i : C_i \subseteq B$ then P' will contain $k + 1$ coalitions, since $\forall i : C_i \setminus B$ will be a non-empty coalition, and we add a new coalition B to P'.

The situation where a deviation of a group of agents can result in any number of coalitions formed is exactly the situation we would like to preclude in our model. Therefore we define the k-blocking coalition with the requirement that the resulting coalition structure maintain its size of exactly k coalitions.

Definition 2. (*k-Blocking coalition*) Let $P = \{C_1, C_2, \dots, C_k\}$ be a coalition structure. A coalition B is *k-blocking* if $\forall a \in B : u(a, B) > u(a, \sigma_P(a))$, and there exists exactly one coalition $C_m \in P$ such that $C_m \subset B$.

We note that Definition 2 is a natural adaptation of the standard definition of a blocking coalition (Definition 1), where any deviation must result in a coalition structure containing exactly k coalitions. Therefore, a deviation is required to take the form of a group of agents who would prefer to join an existing coalition (i.e., C_m). The requirement of exactly one coalition C_m is important: if there are more than one such coalitions, the resulting coalition structure will consist of less than k coalitions. If there is less than one such coalition, the resulting coalition structure will consist of $k + 1$ coalitions. It is only when there is exactly one coalition C_m that the deviation results in an *extension* of an existing coalition (implying that all of the members of the "root" coalition C_m prefer the extension).

Definition 3 *(Weakly k-blocking coalition).* Let $P = \{C_1, C_2, \dots, C_k\}$ be a coalition structure. A coalition B is *weakly k-blocking* if $\forall a \in B : u(a, B) \geq u(a, \sigma_P(a))$, and there is at least one agent $a_i \in B$ strictly benefiting such that $u(a_i, B) > u(a, \sigma_P(a))$. In addition, there exists exactly one coalition $C_m \in P$ such that $C_m \subset B$.

For simplicity, we will refer to a (weakly) k-blocking coalition simply as a (weakly) blocking coalition hereafter.

Definition 4 *(Individually rational).* A coalition structure P is *individually rational* if every agent does at least as well in P as it would do alone, i.e., for all $a \in A, u(a, \sigma_P(a)) \geq u(a, \{a\})$.

We are now ready to define the versions of the core which are suitable for our game, which we call the (strict) k-coalitions-core.

Definition 5 *(k-coalitions-core).* A coalition structure $P \in \Pi_k(A)$ is in the *k-coalitions-core* if P admits no blocking coalition and P is individually rational.

Definition 6 *(strict k-coalitions-core).* A coalition structure $P \in \Pi_k(A)$ is in the *strict k-coalitions-core* if P admits no weakly blocking coalition and P is individually rational.

We note that although we did not allow for a blocking coalition to increase k, we still require that P be individually rational in order to prevent the case where there is an agent with a negative payoff who cannot leave his coalition (and the game).

An example that illustrates the concept of the k-coalitions-core is shown in Fig. 2. Note that $\{C_1, C_2\}$ is a member of the 2-coalitions-core. Indeed, only a_1 or a_2 can possibly deviate to C_2. In that case, the utility of a_1 will decrease by 10 and thus it will not deviate. The utility of a_2 will remain the same and thus it also will not be a member of a blocking coalition. Similarly, $\{C_1', C_2'\}$ is a member of the 2-coalitions-core. However, a strict 2-coalitions-core does not exist in this example, since C_2' is a weakly blocking coalition for the coalition structure $\{C_1, C_2\}$, and C_1 is a weakly blocking coalition for the coalition structure $\{C_1', C_2'\}$. Notice that in this example both coalition structures have a social welfare of 20, which is also the maximal possible social welfare.

We also introduce a weaker stability concept, called group stability, which restricts the type of deviations that can occur. This definition is a natural generalization of inner stability [8], in which there does not exist a blocking that is contained in an existing coalition.

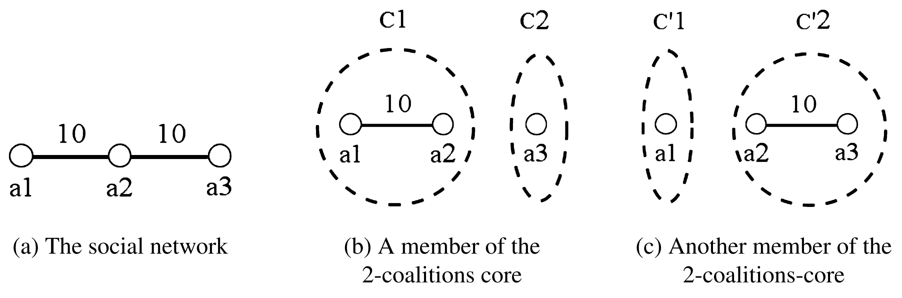

10
10
a1
a2
a3
(a) The social network
C1
C2
10
a1
a2
a3
(b) A member of the 2-coalitions core
C'1
C'2
a1
10
a2
a3
(c) Another member of the 2-coalitions-core

Fig. 2. Members of the 2-coalitions-core, but not its strict version.

Definition 7 *(Group stability).* A coalition structure $P \in \Pi_k(A)$ is *group stable* if P admits no blocking coalition B such that $B \subset C_i \cup C_j$ where $C_i, C_j \in P$ and $i \neq j$.

That is, we only consider the case when a subset of agents from the same coalition benefits from merging into another coalition as a legitimate deviation. This solution concept is motivated by the assumption that it is sometimes only possible for deviating agents to coordinate within their coalition, instead of being able to coordinate across multiple coalitions. We also do not require that P be individually rational, since we want to capture the essence of deviations between coalitions. Note that since B is a blocking coalition, exactly one of C_i or C_j may be a subset of B, as required by Definition 2.

A table summarizing the most frequently used notation from the paper may be found in Appendix A.

3. Maximizing the social welfare

In this section, we investigate the complexity of finding a coalition structure P that maximizes social welfare. We then consider the problem of an organizer who desires to construct a social-welfare maximizing coalition structure to perform k tasks, taking into account the costs incurred by adding edges to the graph.

3.1. Theoretical results

We begin by recalling the following standard graph theoretic definitions.

Definition 8. A *cut* $C = (S_1, S_2)$ of a social network $G = (A, E)$ is a partition of the agents A into two disjoint non-empty sets.[1] Where $(s, t) \in A^2$, we define an *s-t-cut* to be a cut where $s \in S_1$ and $t \in S_2$. A *k-cut* $C_k = (S_1, S_2, \ldots, S_k)$ is simply a partition of A into k disjoint sets, i.e., an element of Π_k.

Definition 9. The *size* of a cut (respectively s-t-cut or k-cut) is defined as the sum of the weights of the edges between each $S_i, S_j \in C$. The Minimum-Cut problem is to find a cut with a minimal size. The Maximum-Cut, Minimum-s-t-Cut and Minimum-k-Cut problems are defined in a similar way.

We denote by Min-Cut, Max-Cut, Min-s-t-Cut and Min-k-Cut the decision versions of Minimum-Cut, Maximum-Cut Minimum-s-t-Cut, and Minimum-k-Cut, respectively.

Brânzei and Larson in [8] refer to the problem of maximizing the social welfare over all possible coalition structures, where we consider maximization only over those of size k. Hence, solutions to these two problems exhibit some similar properties but also several fundamental differences. This will be the subject of discussion later on in this section. Now, as noted by [8], the coalition structure of size k which maximizes the social welfare is the one that minimizes the size of the k-cut. This is true because the sum of all edges is constant, so by minimizing the sum of the edges outside the coalitions, we are maximizing the sum of edges that are within coalitions – in other words, the social welfare. Thus we need to utilize an algorithm for Minimum-k-Cut in order to find the coalition structure that maximizes the social welfare. Note, however, that Min-k-Cut was shown to be NP-complete if the required partition size k is part of the input [23]. Moreover, [16] showed that Min-k-Cut is $W[1]$-hard[2] even if all weights are positive: this implies that the problem is unlikely to admit any algorithm with a complexity of $O(f(k)n^c)$, where f is an arbitrary function and c is a constant. Hence, we assume that k is fixed. Even with a fixed k, Min-k-Cut is still hard when there are edges with negative weights by a simple reduction from Max-Cut which is NP-complete [25]. We show that if the number of edges with negative weights is small, then Min-k-Cut is in P.

[1] Note that in contrast to the common usage of the term "cut", we do *not* require that there be at least one edge in the social network G connecting S_1 to S_2; i.e., it is possible that the sets of agents are not connected. This assumption does not affect our results below.

[2] See [15] for an exact definition and a detailed explanation of this complexity class.

Definition 10 *(Nearly positive graphs).* Let E_{ω^-} be the set of edges in E with negative weights. The cover number $c(G)$ is the smallest size of a node subset X such that each edge of E_{ω^-} has an endpoint in X. We define a class of weighted undirected graphs $\{G_i\}$ as *nearly positive* if $c(G_i) \in O(\log n)$.

For example, if G is a star graph where all the edges are negative, then $c(G) = 1$. If G is a graph where the group of nodes connected by negative edges forms a clique with negative edges, i.e., between every two nodes in the group there is a negative edge, then $c(G) = |\text{clique}| - 1$.

Although the consideration of nearly positive graphs imposes certain limitations, in the context of building cooperating teams for performing k tasks it is reasonable to assume that there should not be too many negative relations in the corresponding social network. Moreover, we did not impose any other restriction on the structure of the social network.

Theorem 1. Min-k-Cut *is in P for nearly positive graphs and a fixed k.*

Proof. Consider Algorithm 3. We generalize the proof of Goldschmidt et al. [23], which showed that Min-k-Cut is in P on graphs with non-negative weights when k is fixed. The essence of their polynomial time algorithm is the recursive application of an algorithm for Minimum-s-t-Cut with multiple sources and sinks. In addition, McCormick et al. [30] showed how to solve the Minimum-s-t-Cut problem in polynomial time for nearly positive graphs. The pseudo-code is given in Algorithm 2, and note that it uses a solution to the Minimum-Negative-s-t-Cut as a subroutine (Algorithm 1). We generalize Algorithm 2 to multiple sources and sinks by adding a super-source node and a super-sink node, connected to the sources and sinks, respectively, where each of their edge weights is $M_\omega = 1 + \sum_{(a_i,a_j)\in E} |\omega(a_i, a_j)|$ (Practically serving as ∞, ensuring that it is best that the nodes be in the same set). Therefore, by replacing the algorithm for Min-s-t-Cut with multiple sources and sinks used by [23] with the generalized version of the algorithm of [30] (Algorithm 3, line 11), we get a polynomial time algorithm for Minimum-k-Cut when k is fixed and the graph belongs to a family of nearly positive graphs, with a complexity of $O(n^{k^2})$ (Algorithm 3). □

Algorithm 1 MINIMUM-NEGATIVE-S-T-CUT(G,s,t).

```
1: Create a graph $G'$ as a duplicate of $G$. Non-existing edges are considered having a weight of 0
2: for all $i$ such that $\omega(s,i) < 0$ or $\omega(i,t) < 0$ do
3:     Set $\omega'(s,i) \leftarrow \omega'(s,i) + |\omega'(s,i)| + |\omega'(i,t)|$ and $\omega'(i,t) \leftarrow \omega'(i,t) + |\omega'(s,i)| + |\omega'(i,t)|$
4: return MINIMUM-S-T-CUT(G',s,t)
```

Algorithm 2 MINIMUM-NEARLY-POSITIVE-S-T-CUT(G,s,t).

```
1: Create a graph $G'$ as a duplicate of $G$. Non-existing edges are considered having a weight of 0
2: Let $M$ be a set cover of the negative edges in $G$.
3: for all Subsets $S$ of $M \setminus \{t\}$ do
4:     Let $T$ be $M \setminus S \cup \{t\}$
5:     Add super node $s'$ and super node $t'$
6:     for all $a_j \notin S \cup T$ do
7:         Set $\omega'(s', a_j) \leftarrow \sum_{a_i \in S} \omega(a_i, a_j)$
8:         Set $\omega'(t', a_j) \leftarrow \sum_{a_i \in T} \omega(a_i, a_j)$
9:     set $\omega'(s', t') \leftarrow \sum_{a_i \in S, a_j \in T} \omega(a_i, a_j)$
10:    Remove all the nodes in $S$,$T$,$s$,$t$ from $G'$
11:    $C_S = (S_1, S_2) \leftarrow$ MINIMUM-NEGATIVE-S-T-CUT($G', s', t'$)
12:    $C_S \leftarrow (S1 \setminus \{s'\} \cup S, S2 \setminus \{t'\} \cup T)$
13: return $\arg\min_{C_S} size(C_S)$
```

Now, since finding a coalition structure that maximizes the social welfare is equivalent to finding the Minimum-k-Cut, we get the following corollary:

Corollary 2. *Finding the coalition structure that maximizes the social welfare can be done in polynomial time if k is fixed and G belongs to a family of nearly positive graphs.*

The strength of this result is that we do not impose any restriction on the graph structure itself. On the other hand, if we want to do away with the assumption of bounded number of coalitions and with the restriction on the number of negative edges we have to impose some restrictions on the structure of the graph. We now show that when the graph is a forest, our problem is solvable in polynomial time, even without the assumption of bounded k and any restriction on the number of negative edges. A forest is an interesting graph to examine in the context of social networks, since it represents a hierarchical structure – a very natural form of organizational social structure. In section 4 we will further consider the restriction to forests with respect to the issue of stability.

Algorithm 3 MINIMUM-NEARLY-POSITIVE-K-CUT(G,k).

```
1: if k=2 then
2:     return  arg min {size(MINIMUM-NEARLY-POSITIVE-S-T-CUT(G,k,s,t))|$s, t \in A$}
3: $w* \leftarrow \infty$
4: for all Subsets $S_1 \subset A$ where $|S_1| = i, i = 1, ..., k-3$ do
5:     Set $w_1 \leftarrow \sum_{a_i \in S, a_j \notin S} \omega(a_i, a_j)$
6:     Let $w_2$ be the optimal size of MINIMUM-NEARLY-POSITIVE-K-CUT($G', k-1$) where $G'$ is induced by $A \setminus S$
7:     if  $w_1 + w_2 < |w*|$ then
8:         $w* \leftarrow w_1 + w_2$ and the corresponding cut is recorded
9: for all subsets $K \subset A$ of size $(k-2) + (k-1) = 2k-3$ do
10:     for all subsets $S \subset K$ of size $k-2$, and $T = K \setminus S$ do
11:         Find $C_{S,T} = (S_1, S_2)$ a MINIMUM-NEARLY-POSITIVE-S-T(G', s',t') where s' and t' are super nodes matching $S$ and $T$
12:         $w_1 = size(C_{S,T})$
13:         Find $w_2$ an optimum size of MINIMUM-NEARLY-POSITIVE-K-CUT($G', k-1$) where $G'$ is induced by $S2$
14:         if $w1 + w2 < w*$ then
15:             $w* \leftarrow w1 + w2$ and the corresponding cut is recorded
16: return  optimal $C = (S_1, S_2, ..., S_k)$
```

Theorem 3. *Finding the coalition structure that maximizes the social welfare can be done in polynomial time if G is a forest.*

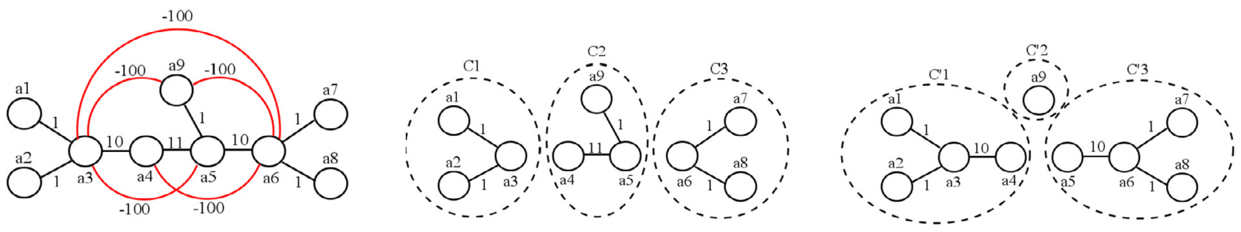
Proof. The intuition behind the proof is that cutting a single edge from a forest always yields two disjoint sets of nodes. We will thus cut all the negative edges (i.e., assign them to the k-cut) and then group all the resulting connected components into k coalitions with no negative edges within the coalitions. However, we need to consider several cases:

- Case 1: there are exactly $k-1$ negative edges. In this case we start with the grand coalition and iteratively increase the number of coalitions in the following manner. In each step we choose a negative edge (a_i, a_j) which belongs to some coalition C to cut. We thus partition C into two new coalitions: C^1, which contains all the reachable nodes from a_i (without the edge (a_i, a_j)), and C^2, which contains the rest of the nodes of C. Note that C^2 may contain nodes that are not reachable from a_j since C may contain disconnected components, and we arbitrarily assigned them to C^2. We perform this step until there are no more negative edges within members of the same coalition. Since there are exactly $k-1$ negative edges we obtain exactly k coalitions, and since our k-cut is minimal we get a coalition structure that maximizes the social welfare.
- Case 2: there are less than $k-1$ negative edges. We start similarly to case 1 and iteratively cut all the negative edges. However, we will now end up with less than k coalitions. We thus iteratively choose a disconnected component $\tilde{C}$ that belong to some coalition C, and partition C into two new coalitions: $\tilde{C}$, and $C \setminus \tilde{C}$ (note that this partition does not change the value of the k-cut). We perform this step until there are k coalitions and we are done. However, if we are out of coalitions that contain disconnected components and there are still less than k coalitions we must add to the cut positive edges. Similarly to case 1, we iteratively cut positive edges, where the edges are selected according to their weight (ascending ordering), until we have k coalitions. As in case 1, since our k-cut is minimal we obtain a coalition structure that maximizes the social welfare.
- Case 3: there are more than $k-1$ negative edges. Here we first perform edge contraction on every positive edge, until we reach a graph containing only negative edges between the merged nodes. Clearly, the resulting graph remains a forest. We then color the nodes of the graph using two colors (which is possible since forests are bipartite graphs and thus 2-COLORABLE). Now we are ready to insert all the negative edges to the cut as done in case 1. However, since there are more than $k-1$ edges we end up with more than k coalitions, and we need to merge them such that there will not be two nodes that are connected with a negative edge within the same coalition. Thus, we iteratively merge two coalitions with the same color, until we end up with k coalitions. Note that the coloring prevents a merge between two nodes that are connected with a negative edge. Moreover, all the positive edges are excluded from the k-cut and thus the resulting coalition structure maximizes the social welfare.

As for the complexity, creating the merged nodes is in $O(|E| \cdot |V|)$, identifying connected components by BFS and the 2-coloring take at most $O(|E| + |V|)$, sorting the edges takes $O(|E| \log |E|)$, and partitioning the nodes into coalitions is in $O((|E| + k) \cdot |V|)$. □

We proceed to examine some aspects of the stability of P^*, the coalition structure that maximizes the social welfare, on general graphs. Note that P^* does not necessarily belong to the k-coalitions-core, as illustrated in Fig. 3. Even when all of the weights are non-negative, P^* is not necessarily a member of the k-coalitions-core. However, it is sometimes natural to assume that the deviating agents cannot coordinate their deviation with all the agents from all the coalitions. Instead, the deviating agents can only coordinate within their coalition. This assumption is defined by the solution concept of group stability, and fortunately P^* has this desirable property, as we show in the following theorem:

Theorem 4. *The coalition structure maximizing the social welfare is group stable.*



(a) The social network (b) The coalitin structure in the 3-coalitions core (c) The coalition struture maximizing SW

Fig. 3. The difference between the 3-coalition-core and the coalition structure that maximizes the social welfare.

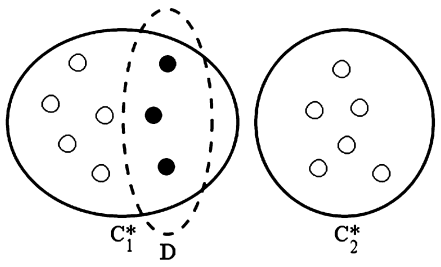


Fig. 4. Illustration of Theorem 4.

Proof. Let $P^* = \{C_1^*, C_2^*, \ldots, C_k^*\}$ be a coalition structure that maximizes the social welfare. Without loss of generality assume there exists $D \subset C_1^*$ such that $C_2^* \cup D$ is a blocking coalition. This is illustrated in Fig. 4. By definition, $\forall a \in C_2^* \cup D : u(a, C_2^* \cup D) > u(a, \sigma_{P^*}(a))$. In particular, this holds for every $d \in D$. Therefore, for a specific $d \in D$:

$$\sum_{a_j \in C_2^*} \omega(d, a_j) + \sum_{a_j \in D} \omega(d, a_j) > \sum_{\substack{a_j \in \\ C_1^* \setminus D}} \omega(d, a_j) + \sum_{a_j \in D} \omega(d, a_j)$$

which implies: $\sum_{a_j \in C_2^*} \omega(d, a_j) > \sum_{a_j \in C_1^* \setminus D} \omega(d, a_j)$

This is true for every $d \in D$, and thus,

$$\sum_{a_j \in C_2^*, d \in D} \omega(d, a_j) > \sum_{a_j \in C_1^* \setminus D, d \in D} \omega(d, a_j) \qquad (*)$$

We calculate the utilities of the coalitions:

$$V(C_1^*) = V(C_1^* \setminus D) + 2 \cdot \sum_{\substack{a_j \in C_1^* \setminus D \\ d \in D}} \omega(d, a_j) + 2 \cdot \sum_{\substack{a_j \in D \\ d \in D}} \omega(d, a_j)$$

$$V(C_2^* \cup D) = V(C_2^*) + 2 \cdot \sum_{\substack{a_j \in C_2^* \\ d \in D}} \omega(d, a_j) + 2 \cdot \sum_{\substack{a_j \in D \\ d \in D}} \omega(d, a_j)$$

Now, since P^* maximizes the social welfare, it holds that: $V(C_1^*) + V(C_2{}^*) \geq V(C_1^* \setminus D) + V(C_2^* \cup D)$, since the rest of the coalitions remain unchanged. Using substitution, we get:

$$0 \leq V(C_1^* \setminus D) + 2 \cdot \sum_{a_j \in C_1^* \setminus D, d \in D} \omega(d, a_j) + 2 \cdot \sum_{a_j \in D, d \in D} \omega(d, a_j) + V(C_2{}^*) - V(C_1^* \setminus D) - V(C_2^*) - 2 \cdot \sum_{a_j \in C_2^*, d \in D} \omega(d, a_j) - 2 \cdot \sum_{a_j \in D, d \in D} \omega(d, a_j).$$

After re-arrangement we get,

$$\sum_{a_j \in C_2^*, d \in D} \omega(d, a_j) \leq \sum_{a_j \in C_1^* \setminus D, d \in D} \omega(d, a_j),$$

which contradicts (*). Therefore, there does not exist any such blocking coalition. □

Corollary 5. *When k equals 2, the coalition structure maximizing the social welfare is a member of the k-coalitions-core.*

Now, we would like to emphasize some inherent differences between our model and that of Brânzei and Larson [8]. In our model, if $P^* = \{C_1^*, C_2^*, \dots, C_k^*\}$ is the coalition structure that maximizes the social welfare, then there might exist some coalition $C_i^* \in P^*$, where a cut of C_i^* is negative. For example, in a clique with negative edges, every coalition structure contains at least one coalition with a negative cut. Therefore, P^* is not guaranteed to be individually rational. Similarly, the k-cut imposed by P^* may contain positive edges. None of these properties holds in the model of [8]. Moreover, if there are only positive edges, the only coalition structure that maximizes the social welfare in the model of [8] is the grand coalition. In our model, in order to enable proper task execution there must be exactly k coalitions, and when k is not fixed finding the coalition structure that maximizes the social welfare is hard even when all of the edges are positive.

We move to address the problem faced by an organizer. This problem has two interconnected aspects. First, it involves improving social welfare by facilitating social relationships between agents, i.e., by adding edges to the social network. Second, it involves finding an optimal partition of the agents. Note that, in general these component problems cannot be considered in isolation: deciding which social relationships to facilitate will in part depend on the partition of agents chosen, and choosing the optimal partition of agents will depend on the social network in place. Thus, the overall problem faced by the organizer is captured by the following optimization problem:

$$\arg\max_{E^{new},P}(SW(P, E^{new}) - \alpha \cdot |E^+|)$$

For this general problem, we identify three cases:

1. *No cost*: $\alpha = 0$. Here, any social relationship can be facilitated at no cost.
2. *Full cost*: $\alpha = 2$. Here, the cost of facilitating a social relationship equals the value of the affinity created.
3. *Intermediate cost*: $0 < \alpha < 2$.

Now, we can immediately see that two of these cases are uninteresting:

Proposition 3.1. *When $\alpha = 0$, the optimal edges to add to G are such that $E^{new} = \{(a_i, a_j)|\, a_i, a_j \in A\} = K_n$, i.e., a complete graph.*

Proposition 3.2. *When $a = 2$, adding edges is not beneficial.*

This leaves the case where $0 < \alpha < 2$. We have:

Theorem 6. *When $0 < \alpha < 2$, the organizer can solve the optimization problem $\arg\max_{E^{new},P}(SW(P, E^{new}) - \alpha \cdot |E^+|)$ in polynomial time if k is fixed and G belongs to a family of nearly positive graphs.*

Algorithm 4 SN-max-SW.

```
1. Set $E^{+'} = \{(a_i, a_j)|\, (a_i, a_j) \notin E\}$ with corresponding weights $\omega'(a_i, a_j) = \frac{2-\alpha}{2}$, and let $E^{new'} = E^{+'} \cup E$.
2. Find the coalition structure $P^*$ in $G' = (A, E^{new'})$ that maximizes the social welfare.
3. Set $E^+ = \{(a_i, a_j)|\, (a_i, a_j) \notin E, (a_i, a_j) \in C, C \in P^*\}$ with corresponding weights $\omega^{new}(a_i, a_j) = 1$, and let $E^{new} = E^+ \cup E$.
4. Return $(E^{new}, P^*)$.
```

Proof. Using Algorithm 4, SN-MAX-SW. Intuitively, the algorithm builds a temporary graph, denoted G', by adding all the possible edges to the original graph (line 1). However, the new edges have adjusted weights in order to simulate the cost incurred to the organizer (α). The algorithm then finds the optimal coalition structure, denoted P^*, in the temporary graph (line 2), which is possible to do in polynomial time according to Corollary 2. The algorithm concludes that the optimal edges to add to G are all the edges that are within a coalition in P^* (line 3), and the coalition structure that maximizes the social welfare in G after adding the edges in E^+ is P^*. In order to prove the correctness, we first show that P^* is the coalition structure that maximizes the social welfare in G, after adding the edges in E^+. Indeed, given a coalition structure P in G', every edge from $E^{+'}$ within a coalition $C \in P$ contributes $2 - \alpha$ to the social welfare. Each such edge is also added to G with a weight of 1 (line 3). By definition, the rest of the edges from $E^{+'}$ do not contribute to the social welfare. Thus, $SW(P, E^{new'}) = SW(P, E^{new}) - \alpha \cdot |E^+|$. In particular, $SW(P^*, E^{new}) - \alpha \cdot |E^+| = SW(P^*, E^{new'}) \geq SW(P, E^{new'}) = SW(P, E^{new}) - \alpha \cdot |E^+|$, for every coalition structure P. That is, P^* maximizes the social welfare in G after adding the edges in E^+, as required. Now we need to show that adding the edges from E^+ (as defined in line 3) is the optimal way of adding edges to G. Indeed, given any set of edges to add to G, $\hat{E^+}$, we can build a temporary graph $\hat{G}'$ with the set of edges $E^{\hat{new}'} = \hat{E^+} \cup E$ where every edge from $\hat{E^+}$ has a corresponding weight of $\frac{2-\alpha}{2}$. Since $\hat{E^+} \subseteq E^+$, by an equivalent argument used in the proof to Theorem 3.1, we get that $SW(P, E^{new'}) \geq SW(P, E^{\hat{new}'})$, for every coalition structure P. In particular,

$SW(P^*, E^{new'}) \geq SW(\hat{P^*}, E^{\hat{new}'})$, where $\hat{P^*}$ is the coalition structure that maximizes the social welfare in G' after adding the edges in $E^{\hat{new}'}$. Therefore, $SW(P^*, E^{new}) - \alpha \cdot |E^+| = SW(P^*, E^{new'}) \geq SW(\hat{P^*}, E^{\hat{new}'}) = SW(\hat{P^*}, E^{\hat{new}}) - \alpha \cdot |\hat{E^+}|$, as required. As for the complexity, lines 1 and 2 take at most $O(n^2)$ steps, and line 3 can be performed in $O(n^{k^2})$ steps, as was shown in Theorem 1. $\square$

We conclude by noting that although Algorithm 4 assumes that the organizer is able to add edges with a weight of 1 and the cost α is between 0 and 2, it can be easily extended, since all the terms in the proof of Theorem 6 are additive. That is, if the organizer is able to add an edge with a weight of ω_1 (instead of 1) and the cost α is between 0 and $2 \cdot \omega_1$, then the adjusted weight $\omega'(a_i, a_j)$ (Algorithm 4, line 1) becomes $\omega'(a_i, a_j) = \frac{2\omega_1 - \alpha}{2}$. In section 3.2 we perform additional experiments to examine this extension.

3.2. Experiments: facilitating relationships on real networks

To better understand the benefits of the adaptations made by the organizer, we sampled data from four different networks:

1. *The Slashdot network*: This is a website where users can tag others as friends/foes (1 and −1 weights, respectively).[3] Since Slashdot's dataset represents a directed network, we defined the undirected edge weight to be the sum of the two directed edges between the nodes. Therefore, the weights of the edges vary between −2 and 2.
2. *The S1 network*: A network of an organization collected from the Facebook pages of the employees. The dataset was collected by Fire et al. [19]. The S1 dataset represents an undirected network, with edge weights representing Facebook friendship: 1 for friendship, 0 otherwise.
3. *Random*: Here we generated random social networks. The graph was generated by assigning weights to all possible edges in a complete graph. Then, all the edges with a weight of 0 were removed from the graph. Overall, the edge weights were generated between −2 and 5. This was achieved by drawing random values from a normal distribution with a mean of zero and a standard deviation of one, then multiplying them by 5. Next, the distribution was adjusted to assign smaller probabilities to negative weights in comparison to positive weights (non-symmetric distribution). All values outside the range of −2 to 5 were mapped to 0 (i.e., the corresponding edge was removed).
4. *Random Kronecker network*: Kronecker graphs were first introduced by Leskovec et al. [29], as a way to model real networks, and are widely used, e.g, [37,33,20]. The method for generating the graphs is by constructing a sequence of graphs from a small initiator matrix (that contains probabilities) by iterating the Kronecker product. We used the matrix $[0.9, 0.5; 0.5, 0.1]$ to generate the graph, simulating a collaboration network (see [29] for the details). Since the graph that is generated is unweighted, we assigned the weights from a normal distribution with a mean of 2 and a standard deviation of 1. However, we also wanted the graph to be nearly positive. Therefore, we uniformly chose roughly $O(\log n)$ nodes, denoted V_-, and for each such node v we uniformly chose a probability p_v. Then, for each edge that is connected to a node $v \in V_-$ the weight was multiplied by −1 with probability p_v, independently.

The details of our experimental setup are as follows:

1. The experiment graphs were randomly sampled from the Slashdot and S1 networks using adjusted BFS. That is, since the original networks are very large, using a standard BFS may result in a star-like graph. We thus limited the number of new neighbors sampled for each node to be one third of the total sampled nodes. For the random network we generated the desired number of nodes. For Kronecker graph, which are generated in sizes that are a power of 2, we generated the closest graph size and used the adjusted BFS to create a sampled graph with the required size.
2. The number of agents varied between 12 and 16, where each sampling was performed 100 times. In addition, we performed sampling with the number of agents set to 10, 20, 30, 40, 50, where each sampling was performed 20 times, to examine the general trend lines.
3. The number of coalitions, k, was set to 3 or 4.
4. α was set to 0, 0.5, 1, 1.5, 2, when the organizer was only allowed to add edges with a weight of 1.
5. In another set of experiments we set α to be 2 and allowed the organizer to add edges with a weight of 2 (instead of 1).

Our goal is to evaluate the performance of Algorithm 4, SN-MAX-SW, for facilitating relationships by adding edges to the network. This was done by examining the relative improvement of the social welfare after the addition of edges. We also tested the number of edges added by the algorithm out of the total number possible.

[3] The network's data was downloaded from snap.stanford.edu.

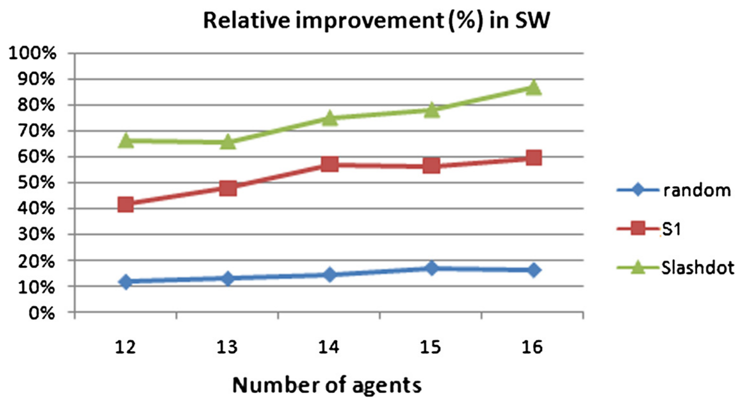


Fig. 5. SW improvement on different networks ($k = 4$, $\alpha = 1$).

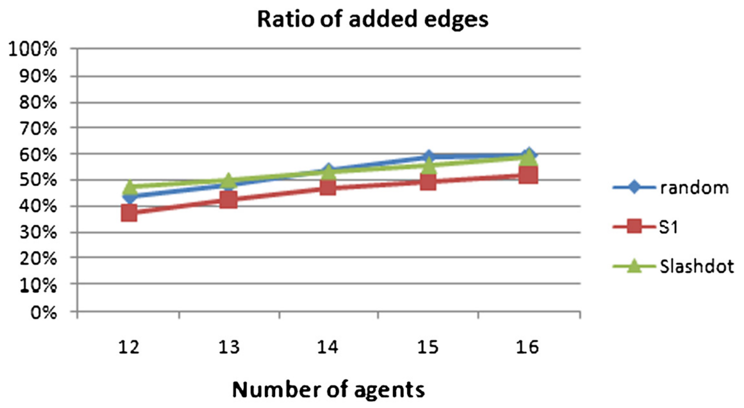


Fig. 6. Ratio of added edges on different networks ($k = 4$, $\alpha = 1$).

3.3. Results

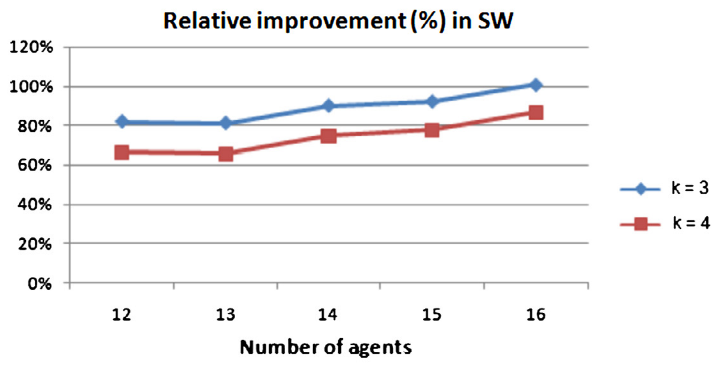
First, we evaluate the improvement with respect to the number of agents. Consider Fig. 5, which visualizes the improvement on different networks. The improvement is most significant for the Slashdot network, followed closely by S1, and by the random network by a large margin. For a different parameter choice, the result is similar. This is due to a difference in scales: Slashdot contains low positive and negative values. Therefore, the addition of new edges with a weight of 1 has a greater effect on the social network. In contrast, the random network contains edges with weights of up to 5, and therefore the improvement is less significant.

To somewhat normalize the effect of scaling we can observe Fig. 6, which considers the number of added edges to the graph. From this we see that the difference between the networks is less significant. These observations indicate that for all networks there is a positive trend line as the number of agents increases. This can be explained by the fact that there are more agents who are non-acquainted as the number of agents increases. Therefore, there is additional potential for improvement of the network, which is illustrated in Fig. 6.

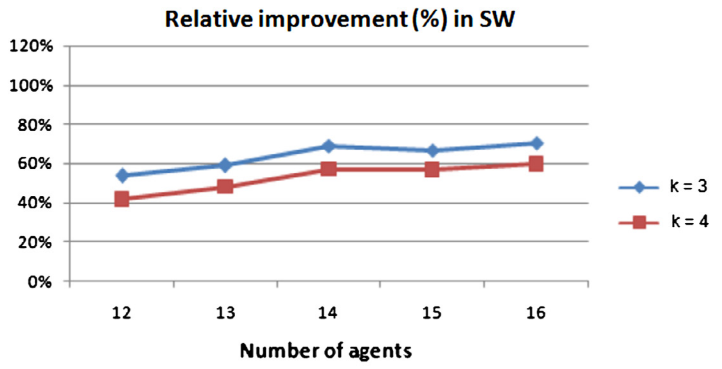
Next we consider the effect of k on the results. In Fig. 7 we compare two different values of k. It seems that the improvement increases as k decreases, which is consistent across all networks. This could be explained as follows: as k decreases, there are fewer partitions in the network, and therefore more relationships can be facilitated.

Next, we examine the effect of the cost of adding edges. Consider Fig. 8, which illustrates the effect on the Slashdot network. Apart from scaling, the behavior (not reported here) is consistent across all networks. When the cost of adding edges is 2 there is no improvement, since each edge's contribution to the social welfare is eliminated by the cost. As could be expected, the improvement increases as the cost decreases. Furthermore, this increase is near linear with respect to the decrease in cost.

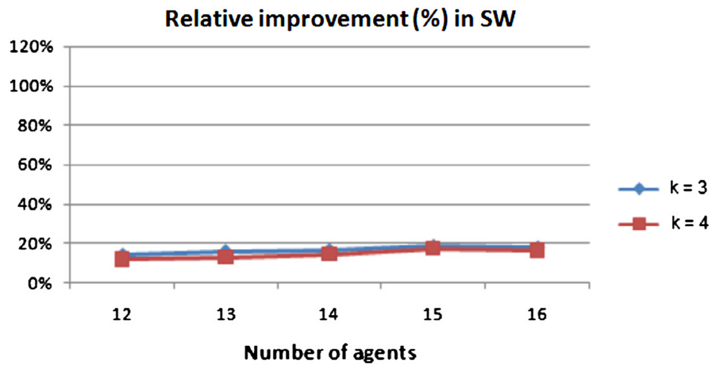
The effect of the cost on the ratio of added edges is illustrated in Fig. 9. The behavior is nearly consistent across networks. We can also observe that for all $0 < \alpha < 2$ the number of added edges is almost the same. That is, the given value of α has almost no effect on the decision of whether to add specific edges or not and therefore on the structure of the resulting graph. Thus, a higher value of α decreases the relative improvement in the social welfare (Fig. 8), but the organizer will still benefit by adding (almost) the same number of edges. In S1, because the weights are either 0 or 1, when the cost of adding edges is 0, it affects the coalition structure significantly compared to the higher edge costs. The graph is continuous up to



(a) Slashdot



(b) S1



(c) Random

Fig. 7. Comparison of the performance with a different number of coalitions. $\alpha = 1$.

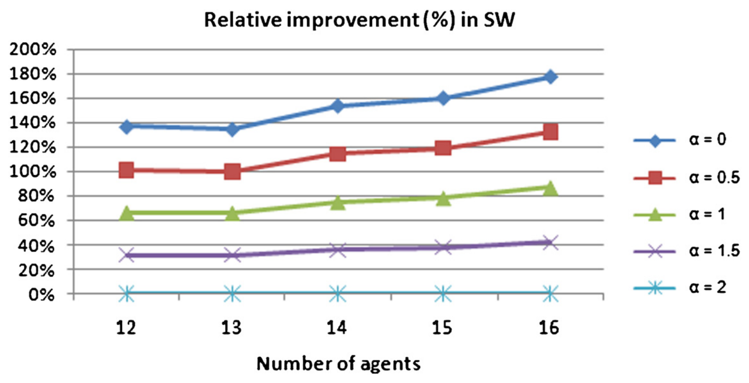


Fig. 8. SW improvement with different edge cost (Slashdot, $k = 4$).

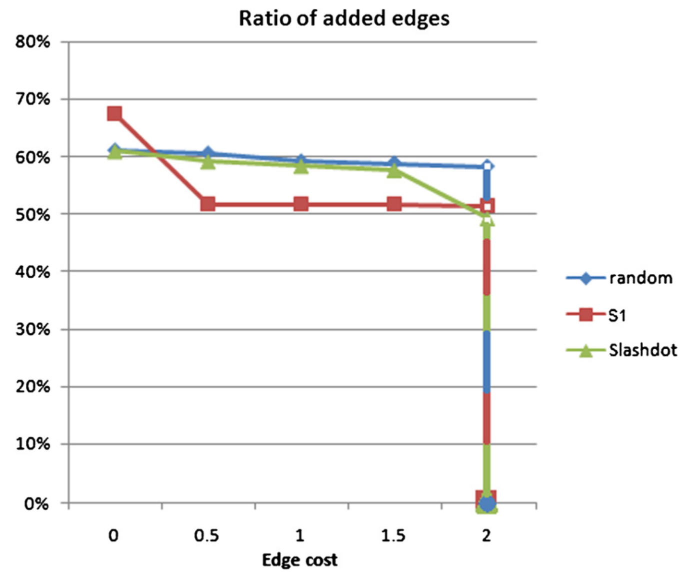


Fig. 9. Ratio of added edges on different networks ($k = 4$, $n = 16$).

Table 1
Average running time for a single graph, in seconds, on the Kronecker network. The time was measured on a single core of i7-3770k CPU.

Cover number \ n	10	20	30	40	50
0	0.06	1.08	12.65	82.69	329.14
2	0.08	2.29	34.03	200.06	803.92
4		8.70	112.91	775.24	3288.43
6				2531.67	11281.19

(a) $k = 3, \alpha = 1$

Cover number \ n	10	20
0	9.81	80926.47
2	18.70	107332.6
4		222690

(b) $k = 4, \alpha = 1$

the point where the cost of adding edges is 2, which is the point of indifference between adding the edges and not adding them.

We proceed to examine the running time and the general trend lines on a larger scale. As can be shown in Table 1, the running time of the algorithm on nearly positive graphs is consistent with the theoretical complexity analysis of $O(n^{k^2})$. Therefore, on the Kronecker network with $k = 3, \alpha = 1$ and a cover number of 4, the average running time on graphs with 50 nodes was less than an hour. When $k = 4$ and the same α and cover number, the average running time was almost 62 hours on graphs with only 20 nodes. We thus set $k = 3$ when we proceeded to examine the improvement in the social welfare on a large scale, as depicted in Fig. 10. We can observe that the improvement is more significant for the Slashdot network than for S1 (as before), and the greatest improvement is for the Kronecker network. Generally, the relative improvement in the SW increases as the number of agents increases, but there are some cases where it decreases. In order to better understand this phenomenon, we depict in Fig. 11 the original optimal social welfare without the added edges. Note that original optimal social welfare is affected by the networks' weights and structure. The S1 network has the highest and monotonically increasing social welfare, as expected from a positive network. The Slashdot network, which contains negative edges but also higher weight scale, follows after. The Kronecker network has the lowest original social welfare values, since it is sparser then the other networks. Now, the correlation between Figs. 10 and 11 shows that a decrease in the original social welfare or a modest increase results in higher relative improvement achieved by our algorithm. Indeed, since the edges added by the organizer have a high impact on the new social welfare when it is originally low, the relative improvement is higher. Thus, when we increase the number of agents the original social welfare increases, which results in a lower relative improvement. On the other hand, with each new agent in the network there are more possibilities to add edges, as we showed before. Hence, Fig. 10 actually shows the combined effect of increasing the number of agents, and it shows that the number of agents has usually a positive effect on the relative improvement of the social welfare by our algorithm.

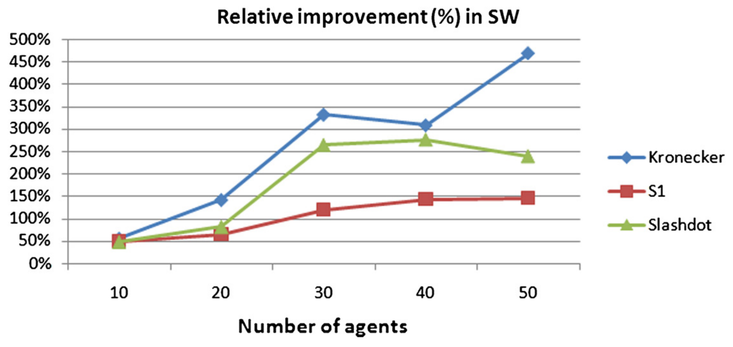


Fig. 10. SW improvement with increasing number of agents ($k = 3$, $\alpha = 1$).

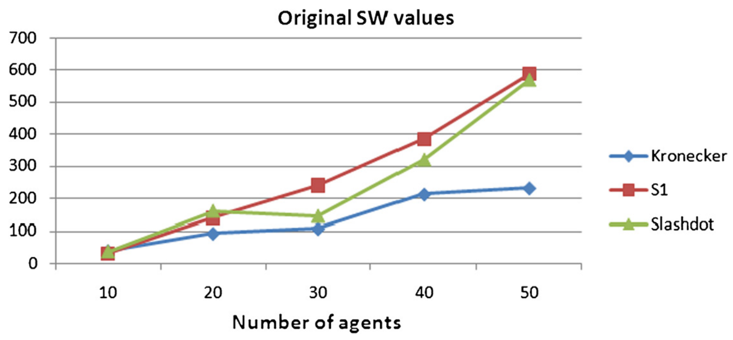


Fig. 11. The original SW (without the added edges) with increasing number of agents ($k = 3$, $\alpha = 1$).

Finally, we turn to evaluate the scenario when the organizer is able to add edges with a weight of 2 and the cost α equals 2. That is, each added edge according to Algorithm 4 contributes 2 to the social welfare. We compare the relative improvement in the SW with the original setting, where the organizer is able to add edges with a weight of 1 and the cost α equals 1, but we did not change the weights of the graph. As Fig. 12 shows, the improvement that is achieved is almost as twice as big. However, if we also increase the scale of the weights in the graph accordingly, i.e., creating a scaled network, Kronecker-scaled, where the mean of the distribution the weights were drawn from was increased by 2 but the structure remains the same, the organizer achieves almost the same performance. We conclude that the weight of the added edges by the organizer is relevant only with respect to the weights of the other edges in the network, and thus all of our results can be easily extended to different edge weights and costs.

In summary, our results illustrate the improvement obtained by applying the Algorithm SN-MAX-SW to real world and synthesized networks. The improvement is consistent, but its magnitude is affected by several parameters:

1. The network's size: usually, the larger the network, the greater the improvement (Fig. 10).
2. The network's structure: the sparser the graph, the greater the improvement (Fig. 10).
3. Number of coalitions: the larger k, the lower the improvement, and the correlation is near linear (Fig. 7).
4. The weight scale: this is the ratio between the original edges' weights in the network and the weight of the edges that the organizer is allowed to add. The larger this ratio, the lower the improvement, and the correlation is near linear (Figs. 5, 12).
5. The cost scale: this is the difference between the weight of the edges the organizer is allowed to add and the cost α. The larger this difference, the greater the improvement, and the correlation is near linear (Fig. 8).

Moreover, the number of added edges out of the possible number of edges to add is almost similar between all networks, and it ranges in our experiments between 40%–60% (Fig. 6). This shows that a naive algorithm, which facilitates all possible relationships, will be far from optimal.

3.4. Greedy heuristic

In Section 3.3 we showed that the running time of the optimal algorithm on nearly positive graphs is consistent with the theoretical complexity analysis of $O(n^{k^2})$, i.e., exponential in k. Moreover, as we have already explained, it is unlikely to find an optimal algorithm with a running time that is polynomial in k. From a practical standpoint, it is still important to find a good coalition structure even where k is not small. Therefore, in this section we propose a greedy heuristic that is polynomial in n and k and test its performance.

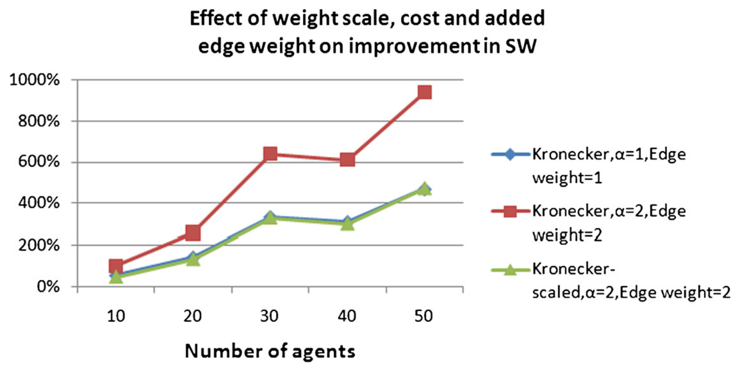


Fig. 12. The linear relationship between the edge weights and cost ($k = 3$).

The heuristic is described by Algorithm 5, and it works as follows. It performs $k - 1$ consecutive cuts, and it thus ends with k coalitions. In each iteration the coalition that yields the minimum cut is greedily chosen to be partitioned into two new coalitions. Since only two new coalitions were created, in the next iteration the heuristic computes two min-cuts of the new coalitions and uses the precomputed cut values for the rest of the coalitions. Specifically, line 1 initializes an array *cut* that stores the min-cut of each coalition. A matching array w is initialized in line 2, to store the sizes of the min-cut for each coalition. *partitioned* is the index of the coalition that was chosen to be partitioned, and it thus initialized to 1. k' represents the current iteration, which is also the current number of coalitions. In the first iteration there is one coalition – the grand coalition (line 4).

Algorithm 5 GREEDY-MINIMUM-NEARLY-POSITIVE-K-CUT(G,k).

```
 1: Initialize $cut \leftarrow$ array of size $k$ with $nil$
 2: Initialize $w \leftarrow$ array of size $k$ with $\infty$
 3: $partitioned \leftarrow 1$
 4: $S_1 \leftarrow A$
 5: for $k' = 1...k-1$ do
 6:     for $i = partitioned$ and $i = k'$ do
 7:         if $|S_i| > 1$ then
 8:             $C' \leftarrow$ MINIMUM-NEARLY-POSITIVE-K-CUT($G'$, 2) where $G'$ is induced by $S_i$
 9:             $cut[i] \leftarrow C'$
10:             $w[i] \leftarrow size(C')$ in $G'$
11:     $partitioned \leftarrow \arg\min_{i=1..k'} w[i]$
12:     cut coalition $S_{partitioned}$ according to the cut stored in $cut[partitioned]$, and create 2 new coalitions: $S_{partitioned}$, $S_{k'+1}$
13:     $cut[partitioned] \leftarrow nil$
14:     $w[partitioned] \leftarrow \infty$
15: return $C = (S_1, S_2, ..., S_k)$
```

In each iteration the heuristic examines the two new coalitions that were created in the previous iteration (line 6), and their indexes are *partitioned* and k'. It computes a min-cut for the two coalitions (using Algorithm 3) and stores it along with its size in *cut* and w, respectively (lines 8–10). Then, the heuristic greedily chooses the coalition with the minimum min-cut size, and partitions it according to the min-cut. There are now two new coalitions; one gets the same index as the original coalition (*partitioned*), and the other gets a new index, $k' + 1$. Note that, after the partition, the computed cut in index *partitioned* is no longer valid (since the index refers to a new coalition), and the heuristic thus resets *cut* and w accordingly (lines 13–14).

Regarding the complexity, the for loop in line 5 executes $k - 1$ times. In each iteration the heuristic computes at most two min-cuts using Algorithm 3, which takes at most $O(n^4)$. Computing the size of the cut takes at most $O(n^2)$, thus the heuristic has a complexity of $O(k \cdot n^4)$, which is linear in k. We can then practically solve the organizer's problem; instead of optimally solving MINIMUM-K-CUT in line 2 of Algorithm 4, we can use the greedy heuristic.

Indeed, we tested the running time of the heuristic on a large scale. We used the Kronecker network with the number of agents varying between $10 - 100$. We set α to $1, 2$ and k varied between 3 and 20. Each sampling was performed 20 times. Fig. 13 shows the running time of the heuristic when we fixed the number of coalitions k and increased the number of agents n. Fig. 14 shows the running time of the heuristic when we fixed the number of agents and increased the number of coalitions. As expected, in both cases the running time was consistent with the theoretical time complexity of $O(k \cdot n^4)$.

We proceed to examine the performance of the heuristic compared to the maximal social welfare that is found by the optimal algorithm. In most of the instances that we checked the heuristic found a coalition structure with the maximal social welfare. Specifically, we used the Slashdot, S1 and Kronecker networks. We set α to $1, 2$ and fixed k to be 3. We varied the number of agents between 10–50. Overall, we tested 600 different instances, and we found the following relative differences between the solution of the heuristic and the maximal social welfare:

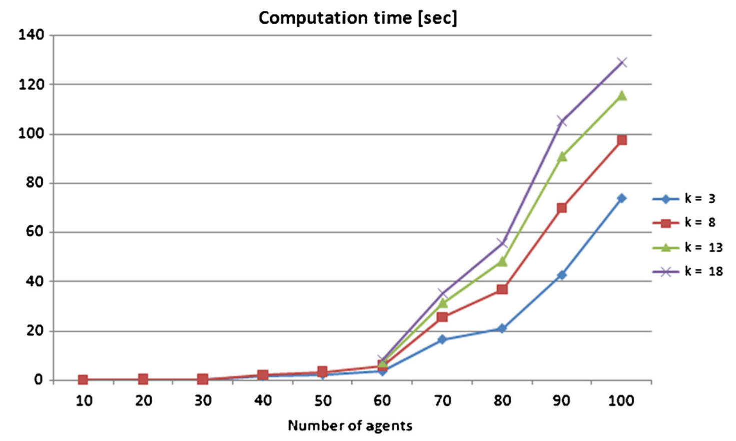


Fig. 13. Computation time of the heuristic with an increasing number of agents.

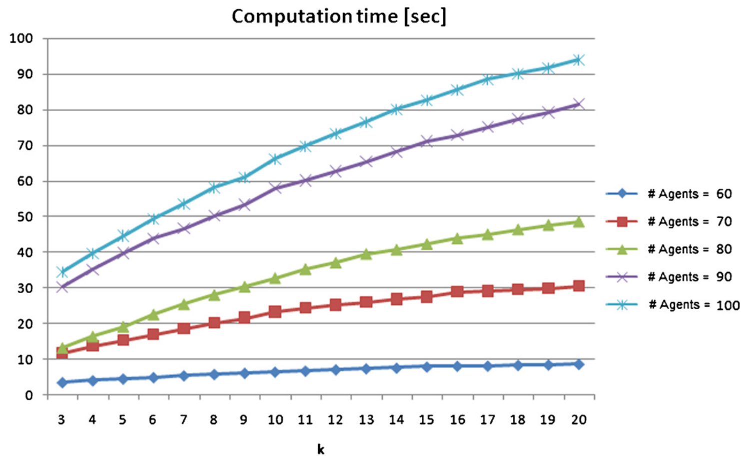
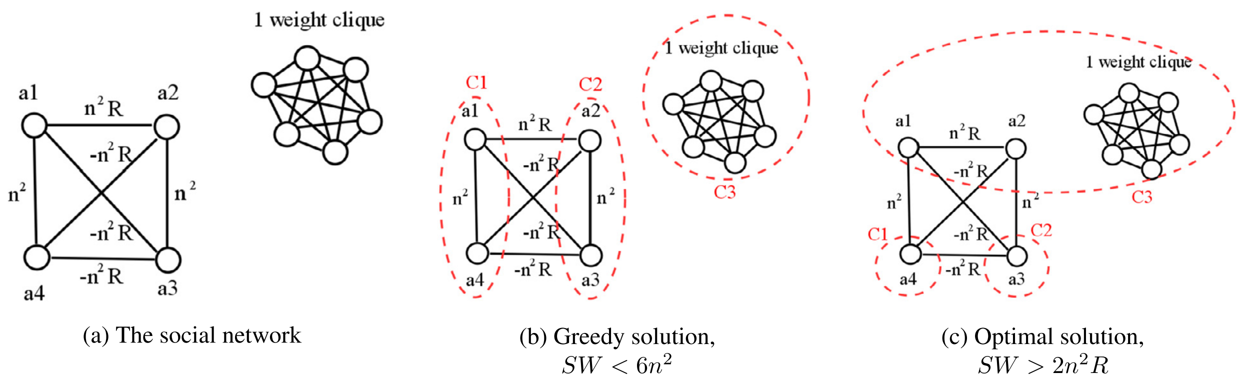


Fig. 14. Computation time of the heuristic with an increasing number of coalitions.

1. s1 – There were 3 instances where the differences were 0.11%, 0.56% and 9%. In all of the other 197 instances the heuristic found a coalition structure with the maximal social welfare.
2. Kronecker – There were 2 instances where the differences were 0.65% and 2.1%. In all of the other 198 instances the heuristic found a coalition structure with the maximal social welfare.
3. Slashdot – There were 6 instances where the differences were 0.36%, 1.80%, 1.90%, 2.70%, 3.10% and 35.50%. In all of the other 194 instances the heuristic found a coalition structure with the maximal social welfare.

It is important to note that even though the heuristic performs well in practice in most cases, we found one case with a noticeable relative difference of 35.5%. Moreover, we show that in general the ratio between the optimal social welfare and the social welfare returned by the heuristic is unbounded. Consider the example in Fig. 15. In this example we set $R > 3$, $k = 3$, and there is a clique of size $n - 4$ where the weight of every edge is 1. Note that this example can be generalized for any $n > 4$ and there will always be only 3 negative edges. Therefore, the graph is nearly positive. Now, due to the negative edges (a_1, a_3), (a_2, a_4), and (a_4, a_3), the first cut of the greedy heuristic will create two coalitions where $a_1, a_4 \in C_1$ and $a_2, a_3 \in C_2$, and the unit edge weight clique will be in one of them. The next cut will create a new coalition that will contain the unit edge weight clique, and we will thus get a social welfare that is less than $6n^2$. However, an optimal solution will put $a_4 \in C_1$, $a_3 \in C_2$, $a_1, a_2 \in C_3$, and the unit edge weight clique in one of them. This coalition structure results in a social welfare greater than $2n^2R$, and thus the ratio between the optimal social welfare and the social welfare of the solution returned by the heuristic is at least $\frac{R}{3}$. Since we can choose R such that $\frac{R}{3}$ is as high as we want, we prove that the ratio is indeed unbounded. Finally, we note that even where of all the edges are positive we can show that the difference between the optimal social welfare and the social welfare returned by the heuristic is unbounded, as demonstrated in Fig. 16.

In summary, our results demonstrate that the greedy heuristic is usable in practice even for large values of n and k, and it can thus find solutions for all instances where the running time of the optimal algorithm is too large. Even though the

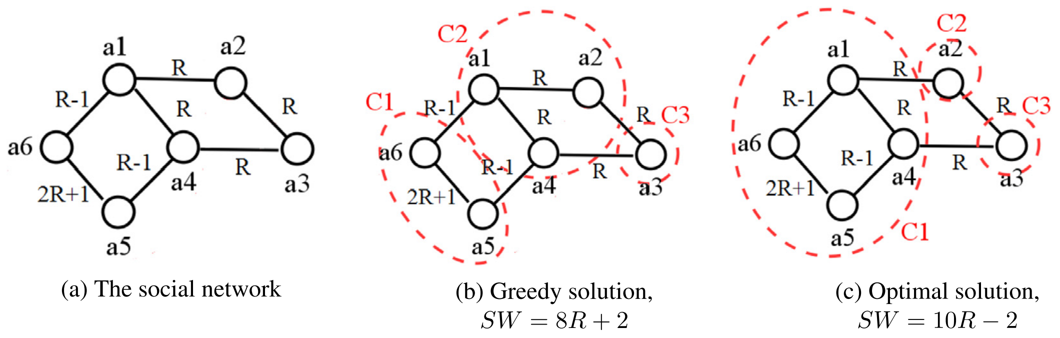


(a) The social network

(b) Greedy solution, $SW < 6n^2$

(c) Optimal solution, $SW > 2n^2R$

Fig. 15. An example of an unbounded ratio between the optimal social welfare and the social welfare returned by the greedy heuristic, $k = 3$.



(a) The social network

(b) Greedy solution, $SW = 8R + 2$

(c) Optimal solution, $SW = 10R - 2$

Fig. 16. An example of an unbounded difference between the optimal social welfare and the social welfare returned by the greedy heuristic on a positive network, $k = 3$.

ratio between the quality of the heuristic solution and the optimal solution is unbounded, the heuristic performs well in practice since in many cases it finds the optimal solution, or a solution that is very close to optimal.

4. The *k*-coalitions-core

4.1. The basic stability problems

We now turn our attention to problems relating to the k-coalitions core. First, let us state the three key decision problems relating to the k-coalitions-core:

Definition 11. K-C-Exist: *Given*: $G = (A, E, \omega)$, the social network (E may contain positive and negative edges), and a natural number k (the number of coalitions). *Question*: Is the k-coalitions-core of G non-empty?

Definition 12. K-C-Membership: *Given*: $G = (A, E, \omega)$, the social network (E may contain positive and negative edges), and a coalition structure $P \in \Pi_k(A)$. *Question*: Is P a member of the k-coalitions-core?

Definition 13. K-C-Find: *Given*: $G = (A, E, \omega)$, the social network (E may contain positive and negative edges), and a natural number k (the number of coalitions). *Output*: A member of the k-coalitions-core, if such exists.

In the analysis of these problems, we make use of the notion of a k-coloring of a graph [12].

Definition 14. A k-coloring of a graph G is a vertex coloring, that is an assignment of one of k possible colors to each vertex of G, such that no two adjacent vertices receive the same color.

Note that a graph may have a proper coloring with less than k colors. In this case we can arbitrarily assign each unused color to a vertex in order to achieve a coloring of exactly k colors. The key decision problem associated with k colorings, which is known to be in NP-complete [21], is as follows:

Definition 15. K-coloring: *Given*: Graph G_c with n_c vertices and natural number k. *Question*: Does G admit a proper vertex coloring with k colors?

We first consider the K-C-EXIST problem. We show that, similar to the existence problem of the core for hedonic games with symmetric and additively separable preferences [3], the problem is NP-hard in the strong sense, even for a fixed k.

Theorem 7. *For a social network with negative and positive weights,* K-C-EXIST *is NP-hard, for every fixed* $k > 2$.

Proof. The reduction is from K-COLORING, when $k = 3$. That is, we prove that the 3-C-EXIST is NP-hard by a reduction from the NP-hard 3-COLORING problem.

Given an instance of 3-COLORING, $G_c = (V_c, E_c)$, we build an instance of 3-C-EXIST such that $A = V_c$, $E = E_c$, and for every $(a_i, a_j) \in E, \omega(a_i, a_j) = -1$. Clearly, this construction can be performed in polynomial time. Now, assume there is a proper 3-coloring, and let $P = \bigcup_i \{C_i\}$, where $C_i = \{a_l |$ the color i assigned to $a_l\}$. We show that P is a member of the 3-coalitions-core. First, it is clear that P consists of exactly 3 coalitions, since the coloring uses 3 colors. None of the vertices with the same color are adjacent (since this is a proper coloring), therefore a vertex does not have any of its immediate neighbors in the same coalition. Hence, the utility of every agent is 0, which is the maximal value possible (since all weights are negative). Therefore, no agent can strictly benefit from moving to a different coalition or by leaving the game, which implies that P is a member of the 3-coalitions-core. For the other direction, assume that there exists a coalition structure P that is a member of the 3-coalitions-core. Moreover, assume that in P there are at least two adjacent vertices, a_i, a_j, in the same coalition. Thus, the utility of a_i and a_j is necessarily negative, and they would benefit from leaving the game. That is, P is not individually rational – a contradiction. Hence, in all of the coalitions of P there are no two adjacent vertices. By assigning the same color to all of the vertices from the same coalition, we get a proper 3-coloring. □

Theorem 7 immediately implies that:

Corollary 8. K-C-FIND *is NP-hard for every fixed* $k > 2$.

We note that K-C-EXIST and K-C-FIND remain hard even if we consider the strict k-coalitions-core, since the hardness of these problems stems from the individual rationality requirement.

We also note that on nearly positive graphs K-C-EXIST is still not trivial, since given a specific k, if G contains a sub-graph that is not k-colorable, then the k-coalitions-core does not exist, regardless of the size of the cover number. The reduction shows that even the question of the existence of a coalition structure with individual rationality is hard. However, even without the requirement of individual rationality the core may not exist, which means that every coalition structure has a blocking coalition. This is illustrated in Fig. 17, where other coalition structures have negative utilities for agents, and therefore are obviously blocked.

We now turn to the K-C-MEMBERSHIP problem. For this problem, we make use of the well-known CLIQUE problem.

Definition 16. CLIQUE: *Input*: Graph $G_c = (V_c, E_c)$ with n_c vertices and natural number s. *Question*: Does G contain a subset of nodes of size s, such that between every two nodes there exists an edge?

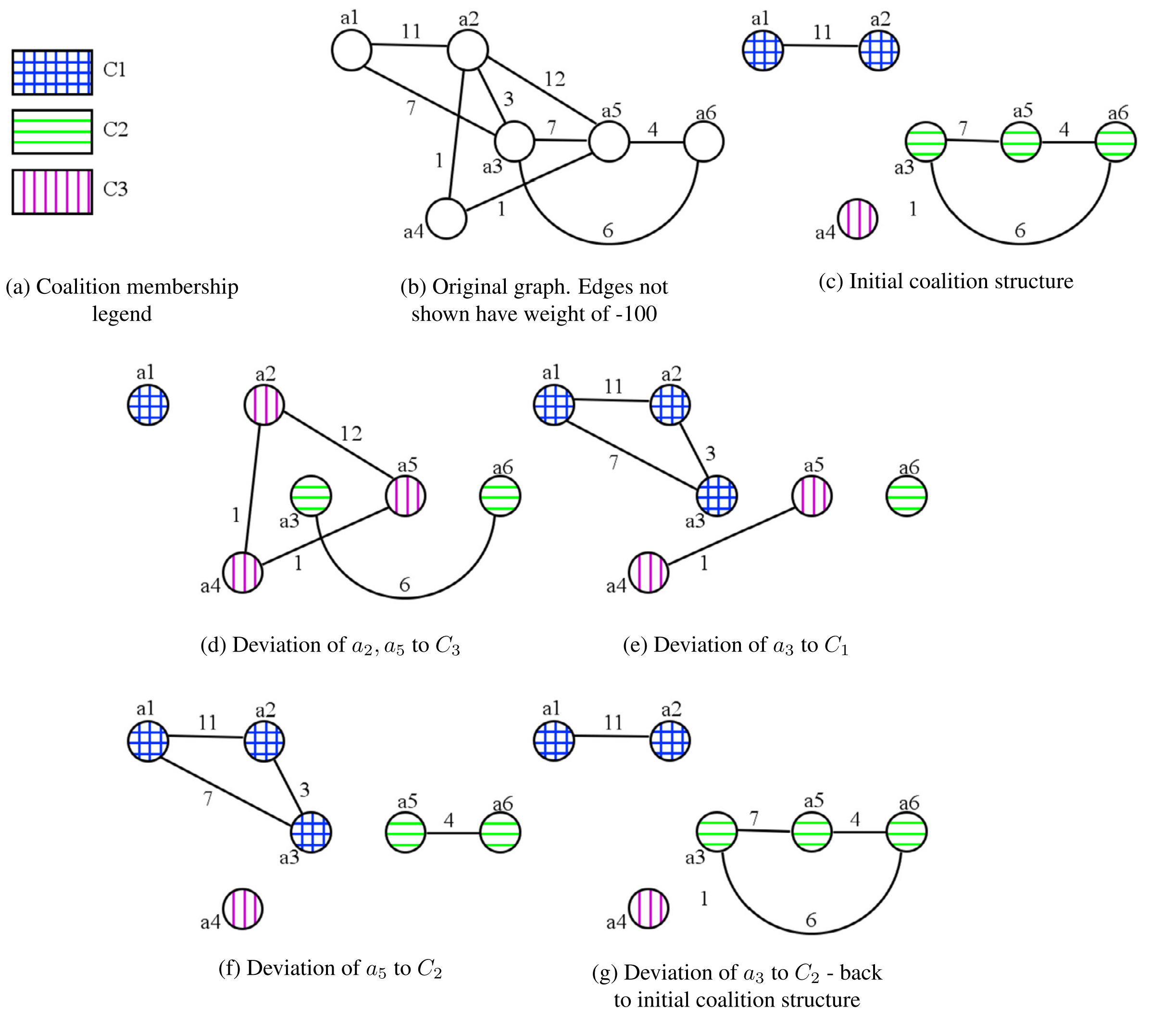
Similar to the CORE-MEMBERSHIP problem in symmetric additively separable hedonic games, which is co-NP-complete [38], the following Theorem establishes that this is also the case with the k-coalitions-core. Below we will examine conditions under which K-C-MEMBERSHIP is in P.

Theorem 9. *For a social network with general negative and positive weights,* K-C-MEMBERSHIP *is co-NP-complete, for every fixed* $k > 2$.

Proof. First we show membership in co-NP by showing that the complement problem, i.e., determining that P is not a member of the k-coalitions-core, is in NP. Indeed, given a deviating subgroup of agents, it is easy to verify that all of the members of the new coalition benefit from the deviation, and that the blocking coalition is valid (i.e., there are no more or less than k coalitions after the deviation).

For hardness, we reduce the CLIQUE problem to the complement of K-C-MEMBERSHIP. Let (G_c, s) be an instance of CLIQUE, where $G_c = (V_c, E_c)$ and $|V_c| = n_c$. We construct an instance of the complement of K-C-MEMBERSHIP as follows. We set $k = 3$, and define $A = V_2 \cup V_3 \cup \{x_1, x_2, x_3\}$, where $V_2 = V_3 = V_c$. That is, A contains two matching nodes for every node in V_c. For a node $i \in V_c$ we denote by a^i_j the matching agent in V_j, for $j \in \{2, 3\}$. We define $P = \{C_1, C_2, C_3\}$ where $C_1 = \{x_1\}$,$C_2 = \{x_2\} \cup V_2$,$C_3 = \{x_3\} \cup V_3$.

We would like to show that P is not a member of the 3-coalitions-core iff there exists a clique Q of size s in G_c. Therefore, we set E such that the blocking coalitions can only be of a specific form which will indicate the existence of a clique, and vice versa. E is defined as follows: x_1, x_2, x_3 are connected by edges of weight $-M_\omega = -(2n_c + 3)^2 \cdot (n_c + s + 1)$, practically serving as $-\infty$, which ensures that they cannot be part of the same coalition, and that each of them will always be a member of a different coalition. That is,



(a) Coalition membership legend

(b) Original graph. Edges not shown have weight of -100

(c) Initial coalition structure

(d) Deviation of a_2, a_5 to C_3

(e) Deviation of a_3 to C_1

(f) Deviation of a_5 to C_2

(g) Deviation of a_3 to C_2 - back to initial coalition structure

Fig. 17. A graph with an empty core, not only due to individual rationality, but also because of a cycle of deviations, which means that every coalition structure has a blocking coalition. Non-relevant coalition structures which are clearly blocked are not shown.

$\omega(x_1, x_2) = -M_\omega,$
$\omega(x_1, x_3) = -M_\omega,$
$\omega(x_2, x_3) = -M_\omega.$

We connect x_1 to all of the agents of V_2 and V_3 with a weight of 1. In addition, we connect x_2 to all of the agents of V_3 with a weight of $-M_\omega$, and respectively x_3 to all of the agents of V_2, to ensure that members of C_2 or C_3 will not deviate to each other's coalitions, and would only want to deviate to C_1. Thus,

$\forall a \in V_2 : \omega(a, x_3) = -M_\omega,$
$\forall a \in V_3 : \omega(a, x_2) = -M_\omega,$
$\forall a \in V_2 \cup V_3 : \omega(a, x_1) = 1.$

In addition, we connect all of the agents from V_2 and V_3 to x_2 and x_3, respectively, with the following weights:

$\forall a \in V_2 : \omega(a, x_2) = n_c + s - 1,$
$\forall a \in V_3 : \omega(a, x_3) = n_c + s - 1.$

The rest of the edges with their corresponding weights connect the agents of V_2 and V_3 so that a blocking coalition may only be C_1 in conjunction with 2 replications of a clique of size s (one from V_2 in C_2 and one from V_3 in C_3):

$\forall (i, j) \notin E_c : \omega(a_2^i, a_3^j) = -M_\omega,$
$\forall (i, j) \in E_c : \omega(a_2^i, a_3^j) = 1,$
$\forall i \in V_c : \omega(a_2^i, a_3^i) = n_c.$

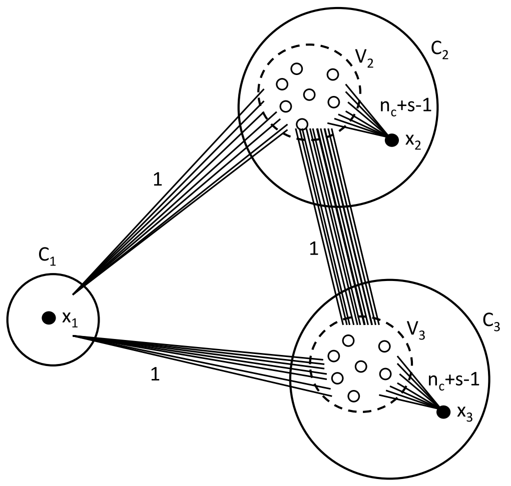


Fig. 18. Illustration of the proof of Theorem 9. Edges with negative weights are not depicted.

Note that there are no edges between agents from V_2 (respectively, between agents from V_3). An illustration of the reduction is depicted in Fig. 18.

Now, assume there exists a clique Q of size s in G_c. We show that P is not a member of the core by finding a blocking coalition, denoted B. We take B to be x_1 with the matching nodes to Q in A, meaning $|B| = 2 \cdot s + 1$. Indeed, every agent in B benefits from the deviation: $u(x_1, B) = 2s > 0 = u(x_1, C_1)$, and $\forall a \in Q, a \neq x_1 : u(a, B) = n_c + s > n_c + s - 1 = u(a, C_i)$, for $i \in \{2, 3\}$. Therefore, P is not a member of the 3-coalitions-core.

For the opposite direction, assume there is a blocking coalition, B. We show that there exists a clique Q in G_c, with $|Q| \geq s$. Clearly, $\{x_1, x_2\} \not\subseteq B$ since $\omega(x_1, x_2) = -M_\omega$. Similarly, $\{x_1, x_3\} \not\subseteq B$, and $\{x_3, x_2\} \not\subseteq B$. In addition, $\forall a \in V_2, \{a, x_3\} \not\subseteq B$ since $\omega(a, x_3) = -M_\omega$. Similarly, $\forall a \in V_3, \{a, x_2\} \not\subseteq B$ Finally, it cannot be that B contains only agents from V_2 and x_1, since $\forall a \in V_2, u(a, V_2) = n_c + s - 1 > 1 = u(a, B)$. Similarly, it cannot be that B contains only agents from V_3 and x_1. From all this, we get that $B = \{x_1\} \cup V_2' \cup V_3'$, where $V_2' \subseteq V_2$ and $V_3' \subseteq V_3$.

Now, we take $Q = \{i | a_3^i \in V_3'\}$. For a given agent $a_2^i \in V_2'$, $u(a_2^i, B) > u(a_2^i, C_2) = n_c + s - 1$, since B is a blocking coalition. Recall that there are no edges between a_2^i and the other agents from V_2, and the only edges with a weight of 1 are either those that connect a_2^i with an agent $a_3^j \in V_3$ such that $(i, j) \in E_c$, or the edge that connects it to x_1. Moreover, the only edge of a_2^i with a weight of n_c is the one that connects it to the matching agent a_3^i. Therefore, $a_3^i \in V_3'$ and there are at least $s - 1$ agents $a_3^j \in V_3'$ such that $(i, j) \in E_c$. In addition, there is no agent $a_3^j \in V_3'$ such that $(i, j) \notin E_c$. Since this holds for all of the agents in V_2', we get that $|V_3'| \geq s$, and for every $i, j \in Q$, $(i, j) \in E_c$. Therefore, Q is a clique of at least size s.

We note that in order to adjust the proof for the case of strict k-coalitions-core, we only change $\omega(a, x_j) = n_c + s, \forall a \in V_j$ for $j \in \{2, 3\}$, in order to ensure that there will be at least s agents for the clique in V_3'. □

Let us now consider conditions under which K-C-Membership is in P. We show that when all the weights are non-negative and k is fixed, it is possible to iterate over all of the potential blocking coalitions in polynomial time.

Theorem 10. *For a social network with only non-negative weights, and k is fixed, then* K-C-Membership *is in P.*

Proof. Consider Algorithm 6. Step (1) verifies that the coalition structure contains exactly k coalitions, otherwise it is easily not a member of the k-coalitions-core. Step (3) is performed in order to ensure that the possible deviating coalition structures are valid in a sense that they have exactly k coalitions. Therefore, we choose k agents, one from each coalition, that are forced to remain in their original coalitions when considering deviations. We consider every such possible selections of k agents.

A possible blocking coalition is a subset of agents deviating to an existing coalition (since there should remain exactly k coalitions). Therefore in step (4) we start iterating over all the coalitions, each time considering a deviation of a subset of agents to a specific coalition.

Now, we need to consider the subset of deviating agents. All agents in the subset must strictly benefit from the deviation, as well as the agents in the coalition that the subset joins. Therefore, in step (6) we initially check a deviation to coalition C_i of all the agents (except the k chosen agents, to ensure k coalitions). Since all weights are non-negative, the utility function of an agent is monotonic in the sense that if an agent does not benefit from a certain coalition, it will not benefit from any of its sub-coalitions. Therefore, if there are some agents in the subset which do not benefit from the current deviation they cannot be a part of the deviating agents (since all other possible blocking coalitions deviating to C_i are sub-coalitions of the current one), therefore we continue to check only the agents that did strictly benefit and that are potentially a part of a blocking coalition. If some agents in C_i do not benefit from the deviation, it means that C_i cannot be a part of the blocking coalition, and we need to consider the next possible coalition to deviate to: C_{i+1}. If all of the agents in the deviation and in coalition C_i strictly benefit from the deviation, then we found a blocking coalition, and therefore P is not a member of the k-coalitions-core. Thus, the algorithm enumerates all possible potentially blocking coalitions for P. If the algorithm did not discover such deviation, then P is guaranteed to be in the k-coalitions-core.

As for the complexity, it is clear that step (1) can be performed in polynomial time. Step (3) contains redundant calculations, but even so, its complexity can be bounded by $\binom{n}{k} = O(n^k)$. Step (4) is performed k times and step (6) is performed at most n times. There are n agents, and calculating the utility for every agent at step (8) takes $O(n)$. In total, we get a complexity of $O(k \cdot n^{k+3})$. Since k is fixed, this is a polynomial time algorithm. □

Algorithm 6 Is-Member-Positive(P,k).

```
1:  if $|P| \neq k$ then
2:     return false
3:  for all possible selection of $k$ agents such that: $x_1 \in C_1, \dots, x_k \in C_k$ $(P = \{C_1, \dots, C_k\})$ do
4:     for $i = 1$ to $k$ do
5:        Initialize $\hat{V} \leftarrow \emptyset$
6:        while $\tilde{V} \leftarrow \{A \setminus (\{x_1, \dots, x_k\} \cup \hat{V})\} \neq \emptyset$ do
7:           $\tilde{C} \leftarrow C_i \cup \tilde{V}$
8:           for every agent $a \in \tilde{C}$ do
9:              calculate $u(a, \tilde{C})$
10:             if $u(a, \tilde{C}) \leq u(a, \sigma_P(a))$ then
11:                if $a \in C_i$ then goto 4
                   else
12:                   $\hat{V} = \hat{V} \cup \{a\}$
13:          if $\forall a \in \tilde{C}$: $u(a, \tilde{C}) > u(a, \sigma_P(a))$ then
14:             return false{$\tilde{C}$ is a blocking coalition}
15: return true{$P$ is in the $k$-coalitions-core}
```

What about the case where we might have a small number of negative edges? As before, we formally capture the notion of a "small number" of negative edges via the family of nearly positive graphs. In the following proof we will use Algorithm 7, which is very similar to Algorithm 6. However, Algorithm 6 utilizes the advantage of having only non-negative weights. In that case, the utility function for every agent has a monotonicity property, in a sense that when an agent does not benefit from a coalition, she will not benefit from any of its sub-coalitions. Therefore, Algorithm 6 does not need to consider all possible sub-coalitions of agents but only a linear number of them. When negative weights are allowed, this property no longer holds true. Agents may well benefit from the removal of other agents with negative edges, thus the utility function is no longer monotonic. Nonetheless, when considering nearly positive graphs, we can still consider all possible subsets of the vertex cover of the negative edges, while using the monotonicity for the rest of the agents. Therefore, Algorithm 7 is able to maintain the polynomial time complexity.

Theorem 11. K-C-MEMBERSHIP *is in P for social networks that are nearly positive graphs and a fixed k.*

Proof. Consider Algorithm 7. Steps(1)–(7) are the same as in Algorithm 6 except for the fact that we now need to test for individual rationality as well. In step (8) we consider every possible subset of the vertex cover of E_{ω^-} (the set of edges in E with negative weights), and use the fact that removing agent $a_i \notin V_-$ will not improve the utility of an agent $a_j \notin V_-$. Indeed, there are three possible cases:

- If $(a_i, a_j) \notin E$ then clearly removing a_i will not affect a_j.
- If $\omega(a_i, a_j) > 0$ then on the contrary, a_j would suffer from the removal of a_i.
- $\omega(a_i, a_j) < 0$ is not possible, because it would mean that either $a_i \in V_-$, or $a_j \in V_-$.

Therefore, when considering a specific subset of V_- we can use the monotonicity on the rest of the agents, as done in Algorithm 6.

Now for the running time, in step (8) we go through all possible subsets of the vertex cover of E_{ω^-}. Since we are not interested in the optimal (i.e., minimal) vertex cover, we can use one of the 2-factor approximation algorithms that finds a vertex cover [28], which still preserves the $O(\log n)$ size of the cover. Enumerating through all possibilities takes $O(2^{\log n}) = O(n)$. Therefore, following a similar argument as in Algorithm 6, we get a total complexity of $O(k \cdot n^{k+4})$. □

As in Section 3, we would like to analyze the complexity of our problem without the restriction on the number of negative edges. We now show that when the graph is a forest and k is fixed, STRICT K-C-MEMBERSHIP, which is the strict k-coalitions-core variant of K-C-MEMBERSHIP, is solvable in polynomial time.

Algorithm 7 Is-Member(P,k).

```
1: if $|P| \neq k$ then
2:     return false
3: if $\exists a \in A$ such that: $u(a, \sigma_P(a)) < 0$ then
4:     return false{$P$ is not individually rational}
5: for all possible selection of $k$ agents such that: $x_1 \in C_1, \ldots, x_k \in C_k$ ($P = \{C_1, \ldots, C_k\}$) do
6:     for $i = 1$ to $k$ do
7:         Initialize $\hat{V} \leftarrow \emptyset$, and $V_- \leftarrow$ vertex cover of $E_{\omega^-}$
8:         for all possible subsets $\hat{V}_-$ of $V_-$ do
9:             while $\tilde{V} \leftarrow A \setminus (\{x_1, \ldots, x_k\} \cup \hat{V}) \neq \emptyset$ do
10:                $\tilde{C} \leftarrow C_i \cup \tilde{V} \cup \{\hat{V}_- \setminus \{x_1, \ldots, x_k\}\}$
11:                for every agent $a \in \tilde{C}$ do
12:                    calculate $u(a, \tilde{C})$
13:                    if $u(a, \tilde{C}) \leq u(a, \sigma_P(a))$ then
14:                        if $a \in C_i$ then goto 6
15:                        else if $a \in \hat{V}_-$ then goto 8
16:                        else then
17:                            $\hat{V} = \hat{V} \cup \{a\}$
18:                if $\forall a \in \tilde{C}$: $u(a, \tilde{C}) > u(a, \sigma_P(a))$ then
19:                    return false{$\tilde{C}$ is a blocking coalition}
20: return true{$P$ is in the $k$-coalitions-core}
```

Theorem 12. STRICT K-C-MEMBERSHIP *is in P when G is a forest and k is fixed.*

Proof. Consider Algorithm 8. Steps (1)–(6) are the same as in Algorithm 7. In step (7) we cut all of the negative edges that are not inside C_i, and we isolate the agents $x_1, x_2, .., x_{i-1}, x_{i+1}, .., x_k$. We then partition the nodes of G into $r + k - 1$ connected component, where the r connected components that do not contain the agents $x_1, x_2, .., x_{i-1}, x_{i+1}, .., x_k$ are denoted by $CC_1, ..., CC_r$. In steps (9)–(10) we iterate over all the connected components CC_h, $1 \leq h \leq r$, in order to check whether $\tilde{C}$, which is the union of C_i with CC_h, contains a blocking coalition. Since the graph now does not have negative edges (except for edges within C_i, but they do not affect the utility of all the agents outside C_i), we can use the same steps of Algorithm 6 to check if $\tilde{C}$ contains a blocking coalition (steps (11)–(20)). Note that we have to add a boolean variable *changed*, which is necessary to preclude an infinite loop in which no agents are excluded from $\tilde{C}$ but there is still no agent that strictly benefits from the deviation $\tilde{C}$.

Clearly, when Algorithm 8 returns true, there exist a blocking coalition and P is not a member of the strict k-coalitions-core. On the other hand, if there exist a blocking coalition then we need to show that the algorithm returns true. Indeed, we show that if there exists a blocking coalition then the algorithm is guaranteed to find a blocking coalition that is a sub-coalition of one of the $CC_h \cup C_i$ that it considers, and it thus returns true. Specifically, we show that if there exists a blocking coalition C' then there exists $1 \leq h \leq r$ and $1 \leq i \leq k$ such that $\tilde{C} = (CC_h \cap C') \cup C_i$ is a blocking coalition.

First note that even after the deviation of the agents of C' we must have k coalitions. Therefore, there is a selection of agents $x_1, x_2, .., x_{i-1}, x_{i+1}, .., x_k$ such that C' does not contain any of them, and we can thus ignore them when we consider the possible $\tilde{C}$. Now, G is a forest, and by removing all the negative edges outside C_i we create a partition of the nodes of G into connected components. Therefore, for every $1 \leq h \leq r$ the utilities of all the agents in $CC_h \cap C'$ are no lower than their utilities in C'. In addition, since C' is a blocking coalition then for some $1 \leq i \leq k$, $C_i \subset C'$ and for some $1 \leq h \leq r$ there is an agent $a \in (CC_h \cap C') \cup C_i$ that strictly benefits from the deviation. Therefore, $\tilde{C} = (CC_h \cap C') \cup C_i$ is a blocking coalition and the algorithm will return true.

As for the complexity, partitioning the graph takes at most $O(n^2)$, and there are at most n connected components. We iterate over all the connected components and in each iteration we use the same steps of Algorithm 6. Therefore, following a similar argument as in Algorithm 6, we get a total complexity of $O(k \cdot n^{k+4})$. □

4.2. Stability and social welfare

We have shown that the coalition structure that maximizes the social welfare is not necessarily a member of the core. In other words, a stable coalition structure may incur a loss in the social welfare. In order to quantify this loss, we use the following measure:

$$Gap_{min}(G, k) = \frac{SW(P^*)}{min_{p \in core(G)} SW(P)}$$

where P^* is a coalition structure maximizing the social welfare.

Brânzei and Larson [8] also examined the bound on Gap_{min}, when there is no limit on the number of coalitions. They proved that when the graph is symmetric with a non-empty core, Gap_{min} is bounded by 2. In our model, it turns out that Gap_{min} is unbounded, as shown by the example in Fig. 19. In this example, a member of the k-coalitions-core has a social welfare of zero, while the coalition structure that maximizes the social welfare has a social welfare of two. Clearly, in this case, Gap_{min} is unbounded.

Algorithm 8 Is-Strict-Member-Forest(P,k).

```
1: if $|P| \neq k$ then
2:     return false
3: if $\exists a \in A$ such that: $u(a, \sigma_P(a)) < 0$ then
4:     return false{$P$ is not individually rational}
5: for all possible selection of $k$ agents such that: $x_1 \in C_1, \ldots, x_k \in C_k$ ($P = \{C_1, \ldots, C_k\}$) do
6:     for $i = 1$ to $k$ do
7:         cut edges $\{(a_j, a_l) | \omega(a_j, a_l) < 0, \{a_j\} \cup \{a_l\} \nsubseteq C_i\}$ and edges $\{(x_j, a_l) | j \neq i, a_l \in A\}$
8:         partition the nodes of $G$ into $r + k - 1$ connected components: $\{CC_1, CC_2, ..., CC_r, x_1, x_2, .., x_{i-1}, x_{i+1}, .., x_k\}$
9:         for $h = 1$ to $r$ do
10:            $\tilde{C} \leftarrow CC_h \cup C_i$
11:            changed ← true
12:            while $\tilde{C} \setminus C_i \neq \emptyset$ and changed do
13:                changed ← false
14:                for every agent $a \in \tilde{C} \setminus C_i$ do
15:                    calculate $u(a, \tilde{C})$
16:                    if $u(a, \tilde{C}) < u(a, \sigma_P(a))$ then
17:                        $\tilde{C} \leftarrow \tilde{C} \setminus \{a\}$
18:                        changed ← true
19:                if $\forall a \in \tilde{C}$: $u(a, \tilde{C}) \geq u(a, \sigma_P(a))$ and $\exists a \in \tilde{C}$ : $u(a, \tilde{C}) > u(a, \sigma_P(a))$ then
20:                    return false{$\tilde{C}$ is a blocking coalition}
21: return true{$P$ is in the $k$-coalitions-core}
```

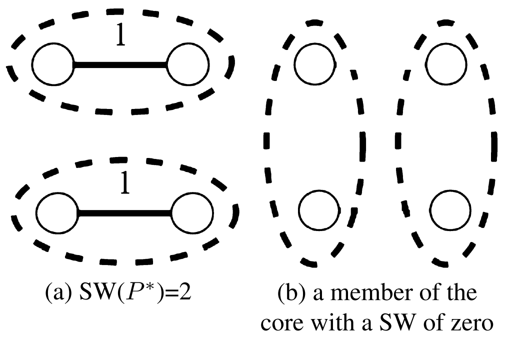


Fig. 19. Unbounded Gap_{min}.

Since finding a member of the k-coalitions-core is proven to be a hard problem, and Gap_{min} is shown to be unbounded, we provide a heuristic for finding a member of the k-coalitions-core that also guarantees a social welfare within a factor of 1/2 of the optimal social welfare. The heuristic is not guaranteed to find a solution, but when it finds a member of the k-coalitions-core we can prove a bound on its social welfare. The heuristic is a straightforward BFS of the following search space:

- State: A coalition structure $P \in \Pi_k(A)$.
- Initial state: A coalition structure maximizing the social welfare, P^*.
- Goal test: A coalition structure that is a member of the k-coalitions-core is found, or the maximal number of iterations is reached.
- Operator: $Deviate(dev, C_i, P) : P \to P'$, which represents a deviation of a group of agents dev, joining coalition $C_i \in P$. Thus, the blocking coalition is $bc = C_i \cup dev$, and $P' = \{C_1 \setminus dev, C_2 \setminus dev, \ldots, C_i \cup dev, \ldots, C_k \setminus dev\}$.
  We place the following constraints on $Deviate$:
  1. No agent can participate in more than one blocking coalition. That is, if the state after t deviations is: $P_t = Deviate(dev_t, C_{i_t}, Deviate(dev_{t-1}, C_{i_{t-1}}, \ldots, Deviate(dev_1, C_{i_1}, P^*), \ldots))$, then for each $j \neq j'$, $bc_j \cap bc_{j'} = \emptyset$, where $bc_j = C_{i_j} \cup dev_j$.
  2. The value of a deviating group within its original coalition is positive. That is, $V(C \cap dev) = \sum_{a,b \in C \cap dev} \omega(a,b) > 0$, $\forall C \in P$.

We note that the heuristic uses BFS as way to search the exponential search space. This yields solutions that are closer to the coalition structure which maximizes the social welfare, i.e., solutions that require the smallest number of deviations from the initial state.

We also note that, although the search space does not contain all of the possible coalition structures, it is still exponential. Nevertheless, using the special deviation operator defined above, we can ensure that the social welfare of the coalition structure found by the heuristic has a social welfare of at least half of the maximal social welfare. The intuition is that a single deviation can affect the utility of only the following relevant agents: those that are part of the blocking coalition

bc, and the neighbors of dev from their original coalitions. Other agents are clearly not affected. The utility of the relevant agents can be reduced at most by a factor of 1/2, since the members of bc get at least as high a utility as before, and the neighbors of dev from the original coalitions get a zero utility out of the agents of dev. Since we require that every deviation use a disjoint set of agents, we ensure that the total social welfare of the coalition structure that is found by the heuristic is still within this bound, as established by the following theorem.

Theorem 13. *The social welfare of the coalition structure returned by the heuristic is at least $\frac{SW^*}{2}$, where SW^* is the maximal social welfare.*

Proof. The heuristic finds the coalition structure that maximizes the social welfare. If it is the member of the core we are done. Otherwise, the heuristics examines the next coalition structure using deviation of blocking coalitions according to BFS. Only agents that were not part of a blocking coalition are considered for future deviations. The heuristic continues until a member of the core is found, or the limit on the number of iterations is reached. In order to simplify the proof, we use the following notations:

- $S(A, B) = \sum_{a\in A}\sum_{b\in B} \omega(a, b)$ – the sum of weights between two sets of nodes, A and B. We use two sums since the set B may depend on the specific $a \in A$.
- $S_a(B) = \sum_{b\in B} \omega(a, b)$ – the sum of weights between a node a and all nodes in the set B.

We begin by proving that the contribution to the social welfare of the edges that are changed by a deviation from one coalition structure, P, to another coalition structure, P', can be lowered at most by a factor of 1/2. Recall the following notations:

- dev – the agents participating in a deviation, leaving their original coalitions.
- $\sigma(a)$ – the coalition of a in the P, meaning $a \in \sigma(a)$.
- C – the coalition into which the agents in dev join.
- bc – the blocking coalition of the deviation, $bc = C \cup dev$.

A deviation may affect several edges. Let Sum^- be the contribution to the social welfare of P of edges that will not contribute to the social welfare of P', i.e., the edges between agents that will **no longer** be members of the same coalition after the deviation. Let Sum^+ be the contribution to the social welfare of P' of edges that are not contributing to the social welfare of P, i.e., the edges between agents that will become members of the same coalition after the deviation. We observe that:

1. $Sum^- = 2 \cdot \sum_{a\in dev} \sum_{b\in\sigma(a)\setminus dev} \omega(a, b) = 2 \cdot S(dev, \sigma(a) \setminus dev)$.
2. $Sum^+ = S(dev, dev \setminus \sigma(a)) + 2 \cdot S(dev, C)$.

In addition, the following constraints must hold:

1. Every agent participating in the deviation must strictly benefit from it:
   $\forall a \in dev$: $S_a(\sigma(a) \setminus dev) < S_a(bc \setminus \sigma(a))$, which in turn equals $S_a(dev \setminus \sigma(a)) + S_a(C)$. Thus, $S_a(\sigma(a) \setminus dev) < S_a(dev \setminus \sigma(a)) + S_a(C)$.
   If we sum over all the agents in dev we get,
   $S(dev, \sigma(a) \setminus dev) < S(dev, dev \setminus \sigma(a)) + S(dev, C)$.
2. Every agent in the coalition into which the agents in dev join must strictly benefit from the deviation:
   $\forall b \in C : S_b(dev) > 0$.
   If we sum over all the agents in C we get,
   $S(dev, C) > 0$

Now, the change in the social welfare due to the deviation is $Sum^+ - Sum^-$. We need to show that $Sum^+ > \frac{1}{2} \cdot Sum^-$. Indeed,

$$Sum^+ - \tfrac{1}{2} \cdot Sum^- = S(dev, dev \setminus \sigma(a)) + 2 \cdot S(dev, C) - S(dev, \sigma(a) \setminus dev).$$

Adding constraint (1), we get:

$$Sum^+ - \tfrac{1}{2} \cdot Sum^- > S(dev, \sigma(a) \setminus dev) - S(dev, C) + 2 \cdot S(dev, C) - S(dev, \sigma(a) \setminus dev) = S(dev, C).$$

However, $S(dev, C) > 0$ according to constraint (2), thus we get that $Sum^+ > \frac{1}{2} \cdot Sum^-$ as required.

Let SW be the social welfare of the coalition structure found by the heuristic after t deviations. According to constraint (1) on the *Deviate* operator, the status of an edge, i.e., whether the edge contributes to the social welfare or not, can be changed once at most. We can therefore write that $SW = Sum_0 + \sum_{i=1}^{t} Sum_i^+$, where Sum_i^+ is the Sum^+ of the i-th deviation, and Sum_0 is the contribution to the social welfare of all the edges whose status was not changed. Note that Sum_0 consists of either edges among agents that have not deviated or edges among agents that have deviated together from the same coalition. The contribution to the social welfare of the former is greater than zero, since the coalition structure that is found is a member of the k-coalitions-core which admits individual rationality. The contribution to the social welfare of the latter is also greater than zero, due to constraint (2) on the $Deviate$ operator which ensures that the utility of a deviating group within its original coalition is positive. Overall, we get that $Sum_0 > 0$, and thus, $SW = Sum_0 + \sum_{i=1}^{t} Sum_i^+ > Sum_0 + \frac{1}{2} \cdot \sum_{i=1}^{t} Sum_i^- > \frac{1}{2} \cdot \sum_{i=0}^{t} Sum_i^- = \frac{SW^*}{2}$.

We conclude by noting that the heuristic can also be used for finding a member of the strict-k-coalitions-core with the same guaranteed bound, and the proof is similar (the constraints on agent utilities just need to be updated to non-strict inequalities). □

To illustrate the performance of the heuristic, we ran simulations on the same 1,800 graphs that we used in the experiments from section 3.2. The experimental parameters were as follows: the number of agents varied between 12 and 16, with a step of 1, and between 10 and 50, with a step of 10. The number of coalitions ranged between 3 and 4, and the maximal number of iterations was limited to 10, 000, which is exponentially less than the entire search space.

In all graphs tested, the core always existed, and the coalition structure maximizing the social welfare was a member of the core. This is consistent with the findings of [8]. In their simulations, the core was always non-empty, the ratio between the maximal social welfare and the maximal social welfare of a core member was very close to 1, and actually in many instances the core contained the social welfare maximizing coalition structure. Even though we have shown that in principle there exist instances where a coalition structure maximizing the social welfare is not a member of the k-coalitions-core, such instances did not in fact appear in our experiments. We hypothesize that this is because the negative weights are not usually structured in the specific way that prevents the coalition structure which maximizes the social welfare from being stable. Indeed, we conjecture that if all of the weights are positive then the core will always be not empty and a coalition structure which maximizes the social welfare will be a member of the core.

In contrast to the case of k coalitions core, in the case of strict k coalitions core, the coalition structure maximizing the social welfare produced by the algorithm was not always a member of the core. The results of running the heuristic are as follows:

- *Slashdot*: A member of the strict k-coalitions-core was found in all of the cases. Moreover, even though in 16.66% of the cases the initial coalition structure was not a member of the core, the core member that was found did not maximize the social welfare in only 0.166% of the cases, and even then the difference was by less than 5%. The average number of iterations was 18 steps.
- *S1*: A member of the strict k-coalitions-core was found in all of the cases. Moreover, even though in 9.16% of the cases the initial coalition structure was not a member of the core, the heuristic always found a core member with the maximal social welfare. The average number of iterations was 5 steps.
- *Random*: In 97% of the cases a member of the strict k-coalitions-core was found within 10, 000 iterations. In 11% of the cases the initial coalition structure was not a member of the core, but if a member of the core was found it also had the maximal social welfare. The average number of iterations was 36 steps.
- *Kronecker*: The initial coalition structure was a member of the strict k-coalitions-core in all of the cases.

4.3. Facilitating relationships

Consider an organizer that would like to add edges to the social network, in order to increase the social welfare of the k-coalitions-core. While adding edges to the social network causes a linear increase in the maximal social welfare, the impact on the k-coalitions-core is more complex. The addition of edges may cause a non-linear decrease or increase in the social welfare of the k-coalitions-core members, even without considering the cost of adding edges (α). This phenomenon is illustrated in Fig. 20. The original 3-coalitions-core member in 20(d) has a social welfare of 20, while the new 3-coalitions-core member after adding all of the edges has a social welfare of only 15.

A more interesting perspective will be to facilitate relationships in order to obtain stability. That is, the organizer needs to decide what the minimal number of edges to add to the network is in order to make a certain coalition structure stable. Adding edges to the social network in order to stabilize it is closely related to the cost of stability, which has been studied mainly in the context of transferable utility games, e.g., [6,31]. In these games the problem is to stabilize a coalitional game by using external payments. A supplemental payment to agents in a specific coalition structure is given in order to persuade agents not to deviate and form a blocking coalition. The cost of stability is then the minimal external payment which stabilizes the coalition structure. In our model the external payment is modeled as facilitating relationships between agents, which will persuade them to work together and not deviate. As opposed to external payment, facilitating relationships as a type of payment has its limits: the organizer can only facilitate new relationships, and each such relationship adds the same weight to two agents.

We note that although it is easy to determine the required payment for every agent when examined separately, this does not yield the correct minimal payment required to stabilize the coalition structure as a whole (that is, the number of

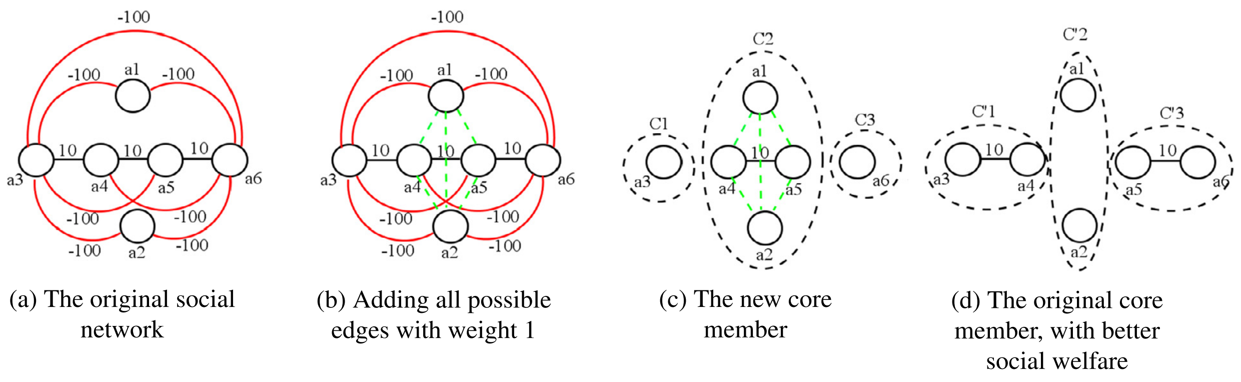

-100
-100
a1
-100
10
10
10
a3
a4
a5
a6
-100
-100
-100
a2
-100
(a) The original social network
(b) Adding all possible edges with weight 1
C1
C2
C3
(c) The new core member
C'1
C'2
C'3
(d) The original core member, with better social welfare

Fig. 20. The effect of adding edges on the k-coalitions-core.

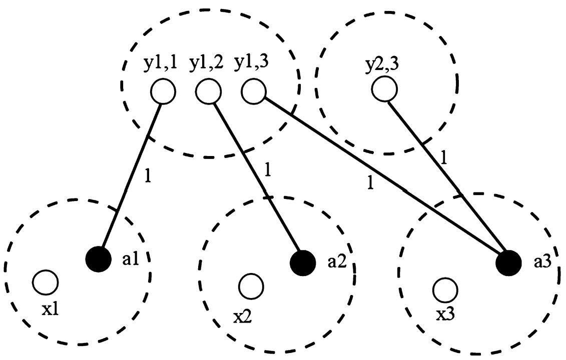


Fig. 21. An illustration of the K-C-Add instance, where $D = \{d_1, d_2\}, d_1 = \{1, 2, 3\}, d_2 = \{3\}$.

edges the organizer needs to add to each agent). This is because the incentives for agents to deviate in a blocking coalition are dependent. It is sufficient to eliminate the incentive of a single agent in the blocking coalition in order to eliminate the blocking coalition altogether. An algorithm for finding the minimal number of edges to add in order to stabilize a specific coalition structure needs to examine all agent incentives together, which unfortunately causes the problem to be NP-hard.

Definition 17. K-C-ADD: *Given*: $G = (A, E, \omega)$, the social network (E may contain positive and negative edges), a coalition structure $P \in \Pi_k(A)$ (that contains exactly k coalitions), and a natural number q. *Question*: Can at most q edges be added to the social network such that P is a member of the k-coalitions-core?

We will show that K-C-ADD is hard, by a reduction from the known NP-hard problem, HITTING-SET [25].

Definition 18. HITTING-SET: *Given*: A collection of subsets $D = \{d_1, \dots, d_n\}$ of a finite subset S, and a natural number h. *Question*: Does there exist a hitting-set for D of size of at most h, i.e, a subset $S' \subseteq S$ such that S' contains at least one element from each subset in D, where $|S'| \le h$?

Theorem 14. K-C-ADD *is NP-hard.*

Proof. We are given an instance of the HITTING-SET problem: (D, S, h). We construct an instance of the K-C-ADD in the following manner: $G = (A, E, \omega)$, where $A = A_d \cup X \cup Y$: $A_d = \{a_j | j \in d_i, i = 1 \dots n\}$, $X = \{x_j | j \in d_i, i = 1 \dots n\}$ and $Y = \{y_{i,j} | j \in d_i, i = 1 \dots n\}$. This means that X and A_d are sets of agents representing all the elements that are members of at least one subset d_i. In contrast, Y contains replicas of agents for each subset of elements d_i. E is defined as follows: $\forall y_{i,j} \in Y, a_j \in A_d : \omega(y_{i,j}, a_j) = 1$, and we set q to be h. Finally, we define the coalition structure P to be: $\bigcup_{j=1\dots|A_d|} \{\{a_j, x_j\}\} \cup \bigcup_{i=1\dots n} \{\{y_{i,j} | j \in d_i\}\}$, i.e. k equals $|A_d| + |D|$. This is illustrated in an example in Fig. 21, for $D = \{d_1, d_2\}, d_1 = \{1, 2, 3\}, d_2 = \{3\}$. We show that the instance of K-C-ADD has a solution if and only if the instance of HITTING-SET has a solution.

We observe that there are exactly $|D|$ blocking coalitions: for each $i = 1 \dots n$, there is a blocking coalition $\{a_j, y_{i,j} | j \in d_i\}$. Note that all of the agents a_j such that $j \in d_i$ must be members of the blocking coalition, since otherwise there will be agents $y_{i,j}$ that will not benefit from the deviation. In addition, such a blocking coalition cannot contain more members, since there are no other agents that will benefit from this deviation. Deviations are possible only to the coalitions of the form $\{y_{i,j} | j \in d_i\}$, since other coalitions contain agents from X which do not have edges to other agents and therefore will not benefit from the deviation. Therefore, since there are exactly $|D|$ blocking coalitions, which correspond to the elements of D, in order to stabilize the coalition structure P the organizer needs to add edges to the possibly deviating agents within their original coalitions, which means edges between agents a_j and x_j.

Now, assume the organizer can stabilize P with at most h edges. Therefore, there is a set of agents $A_h \subseteq A_d$, $|A_h| \leq h$, that are covered by new edges which eliminate their incentive to deviate. Since the resulting coalition structure is a member of the k-coalitions-core, it means that all $|D|$ blocking coalitions have been eliminated. That is, at least one agent from each blocking coalition no longer benefits from the deviation and therefore $S' = \{j | a_j \in A_h\}$ is a valid hitting set of size at most h.

From the opposite direction, assume the organizer cannot add at most h edges to make the coalition structure stable. Therefore, for any addition of at most h edges between agents a_j and x_j (the only way to eliminate deviation incentive) there is at least one blocking coalition whose all members still have an incentive to deviate, meaning none of them was covered by a new edge. Since a blocking coalition is of the form: $\{a_j, y_{i,j} | j \in d_i\}$, it means that $\exists i$ such that none of the elements of d_i is covered. Therefore, there is not a hitting set of size at most h. $\square$

Note that the proof of Theorem 14 can be easily generalized for the case where the organizer is able to add edges with a fixed weight of ω_1. The only change in the reduction is that we set $\omega(y_{i,j}, a_j) = \omega_1$ instead of 1.

5. Related work

Coalition formation in the context of task allocations has been the subject of multiple studies in different disciplines [14,2,34,5]. Some have modeled coalition formation using transferable utilities [34], but it is also very common to assume, as in our model, that the utilities are non-transferable [17].

Our model is a special case of symmetric additively separable hedonic games (symmetric ASHGs), which have been extensively studied in the field of multi-agent systems [10,3]. However, we use the interpersonal relationships, as described by a social network, to define the utilities of the players in the coalition formation game. Brânzei and Larson [8] and Bachrach et al. [7] also defined the utilities is such a way. However, in [7] Bachrach et al. imposed some constraints on the structure of the social network, (such as planar, minor free and bounded degree graphs), in order to obtain constant factor approximations for the problem of maximizing the social welfare. We provide an exact algorithm using a constraint on the number of negative edges in the social network, but not on its structure. We also provide an exact algorithm without any constraint on the weight of the edges, where the social network is a forest. Brânzei and Larson [8] investigated properties of the coalition structure which maximizes the social welfare, the core, and the relationship between them, but did not provide algorithms to find them. In other work [9], Brânzei and Larson used the social network to define the utilities of players, but in a different way. A recently introduced model by Aziz et al. [4] of fractional hedonic games defines the utility of each agent as an average of the values an agent ascribes to the rest of the agents within its coalition. Voice et al. [39] proposed a generalization of the utility model we use, which they term the independence of disconnected members. They provided bounds on the complexity of maximizing the social welfare over general and minor-free graphs and derive linear time bounds for graphs of bounded treewidth.

None of these works considered the possible requirement of forming exactly k coalitions. As we discussed in Section 3, this restriction adds some unique properties to the coalition structure which maximizes the social welfare, and we also had to introduce a modification of the well-known core solution concept, which we called the k-coalitions-core. A similar model of group formation based on the assumption that individuals prefer to associate with people similar to them was studied by Milchtaich and Winter [32]. They also place an upper bound (though not exact) on the number of coalitions to prevent a trivial solution. In contrast to our work, their utility model is based on the distance between agents. Thereby, without the constraint on the number of coalitions, agents would prefer being alone or members of a homogeneous coalition. In addition, their proposed stability concept is Nash-stability, meaning only individual deviations are considered, which is less strong than the core concept.

In another branch of work, [2], [14] and [11] used a social network in coalition formation games as a constraint on possible collaborations, e.g., each coalition must form a connected component. In our model we do not use this constraint, since the organizer has the ability to facilitate relationships and make introductions between unacquainted agents. Similar to the problem that we have defined for the organizer who would like to improve social welfare by adding edges, the work of [22] considers a problem of rewiring the network to enable better team formation in agent organized networks (AON). The key difference is that in their model each agent is able to rewire its local neighborhood, while in our case there is a centralized organizer who can add edges throughout the entire network. Apart from applying local strategies, they also consider multiple iterations, as the agents adapt to the ongoing changes of the network. This direction is also pursued by [24] who consider the re-organization of agents by using temporary edge links between the deviations. We have also defined the K-C-Add problem, for the organizer who would like to stabilize a given coalition structure by adding edges. In the domain of network bargaining games [26] there were also works on stabilizing a given instance of the game, but they investigate the ability to remove agents [1] or edges of the graph [27].

6. Conclusions and future work

We analyzed the problem of coalition formation with a fixed number of coalitions where, in addition, a central organizer can facilitate new relationships between agents at a certain cost. We examined this scenario from two aspects: maximizing social welfare and finding core stable solutions. We provided general results, and established that the problem of finding a coalition structure maximizing the social welfare is tractable only when both k and the number of negative edges are constrained. We provided a polynomial time algorithm in the case of fixed k and nearly positive graphs for computing the optimal coalition structure and the optimal set of edges the organizer should add to the network. We conducted experiments on 4 sample networks which clearly show the benefits of the proposed algorithm. To further improve the running time of the optimal algorithm, we provided an efficient greedy heuristic. We showed that even though the performance of the heuristic is unbounded in theory, it is near optimal in practice.

With respect to core stable solutions, we identified tractable instances and provided polynomial time algorithms for them. When a polynomial time algorithm was not available we provided a heuristic for finding a core stable solution. We also showed that stability may incur a cost in social welfare and that this ratio may be unbounded. We then proceeded to characterize the problem of facilitating relationships in order to stabilize a coalition structure which is closely related to the cost of stability problem. Unfortunately this problem proves to be NP-hard.

We foresee a variety of possible extensions of our work. First, it would be interesting to consider additional task related aspects. For example, suppose that there are different skills for the agents, and each task requires a specific subset of skills to be completed. The combination of a task aspect together with personal relationships will target a very realistic real world scenario [13]. The challenge would be to define a model that captures all of these aspects without increasing the computational complexity. Another potential direction is to relax our requirement of exactly k coalitions and consider coalition formation problems where the coalition structure contains **at least** k coalitions. To eliminate trivial solutions, it is possible that a constraint on the minimal size of a coalition will need to be added. Lastly, we showed that the k-coalitions-core may be empty if the social network contains positive and negative edges (see Fig. 17). We conjecture that the k-coalitions-core is always non-empty if the social network does not contain negative edges, but this remains an open question.

Acknowledgements

Sarit Kraus and Noam Hazon would like to acknowledge the support of the Israel Science Foundation (grant No. 1488/14).

Sarit Kraus would like to acknowledge the support by the Ministry of Science and Technology, Israel and the Japan Science and Technology Agency (JST), Japan. Michael Wooldridge would like to acknowledge the support of the European Research Council under Advanced Grant 291528 (“RACE”).

Appendix A. Notational conventions

A	The set of agents. Members are typically denoted by a_i,a_j. Subsets of A, e.g. coalitions are denoted by C_1,C_2, etc.
E	The set of edges represent the relationships between the set of agents.
ω	The weight function on E representing the relationship strength. $\omega(a_i, a_j) = \omega(a_j, a_i) \in \mathbb{Z}$.
G	An undirected weighted graph representing the social network between the agents. $G = (A, E, \omega)$.
$u(a_i, C)$	The utility of agent a_i in Coalition C, which equals the sum of its edges to other agents within the coalition, $\sum\limits_{a_j \in C} \omega(a_i, a_j)$.
$V(C)$	The value of a coalition, which is defined to be $\sum_{a_i \in C} u(a_i, C)$
k	The number of tasks needed to be performed.
$\Pi_k(A)$	The set of partitions of agents of A that contains exactly k non-empty subsets.
P	A coalition structure. $P \in \Pi_k(A)$.
$\sigma_P(a)$	A mapping $\sigma_P : A \to P$ from an agents to its coalition in the coalition structure P. If P is clear from the context, we will omit it.
$SW(P)$	The social welfare of a coalition structure P.
P^*	A coalition structure that maximizes the social welfare. We will sometimes denote the maximal social welfare by SW^*.

References

[1] S. Ahmadian, H. Hosseinzadeh, L. Sanità, Stabilizing network bargaining games by blocking players, in: International Conference on Integer Programming and Combinatorial Optimization, 2016, pp. 164–177.

[2] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, S. Leonardi, Online team formation in social networks, in: Proceedings of the 21st International Conference on World Wide Web, WWW-12, 2012, pp. 839–848.

[3] H. Aziz, F. Brandt, H.G. Seedig, Computing desirable partitions in additively separable hedonic games, Artif. Intell. 195 (2013) 316–334.

[4] H. Aziz, F. Brandt, P. Harrenstein, Fractional hedonic games, in: Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS-2014, 2014, pp. 5–12.

[5] Y. Bachrach, J.S. Rosenschein, Coalitional skill games, in: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS-2008, 2008, pp. 1023–1030.
[6] Y. Bachrach, E. Elkind, R. Meir, D. Pasechnik, M. Zuckerman, J. Rothe, J.S. Rosenschein, The cost of stability in coalitional games, in: Algorithmic Game Theory, in: Lecture Notes in Computer Science, vol. 5814, Springer, 2009, pp. 122–134.
[7] Y. Bachrach, P. Kohli, V. Kolmogorov, M. Zadimoghaddam, Optimal coalition structure generation in cooperative graph games, in: Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, AAAI-2013, 2013, pp. 81–87.
[8] S. Brânzei, K. Larson, Coalitional affinity games, in: Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS-2009, 2009, pp. 1319–1320.
[9] S. Brânzei, K. Larson, Social distance games, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI-2011, 2011, pp. 91–96.
[10] G. Chalkiadakis, E. Elkind, M. Wooldridge, Computational Aspects of Cooperative Game Theory, Morgan–Claypool, 2011.
[11] G. Chalkiadakis, E. Markakis, N.R. Jennings, Coalitional stability in structured environments, in: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, AAMAS-2012, 2012, pp. 779–786.
[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, The MIT Press, Cambridge, MA, 1990.
[13] C.K.W. De Dreu, L.R. Weingart, Task versus relationship conflict, team performance, and team member satisfaction: a meta-analysis, J. Appl. Psychol. 88 (4) (2003) 741–749.
[14] M.M. de Weerdt, Y. Zhang, T. Klos, Multiagent task allocation in social networks, Auton. Agents Multi-Agent Syst. 25 (1) (2012) 46–86.
[15] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer Science & Business, Media, 2012.
[16] R.G. Downey, V. Estivill-Castro, M. Fellows, E. Prieto, F.A. Rosamond, Cutting up is hard to do: the parameterized complexity of k-cut and related problems, Electron. Notes Theor. Comput. Sci. 78 (2003) 209–222.
[17] P.E. Dunne, S. Kraus, E. Manisterski, M. Wooldridge, Solving coalitional resource games, Artif. Intell. 174 (1) (2010) 20–50.
[18] E. Elkind, M. Wooldridge, Hedonic coalition nets, in: Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS-2009, 2009, pp. 417–424.
[19] M. Fire, R. Puzis, Organization mining using online social networks, Netw. Spat. Econ. (2015) 1–34.
[20] M.G. Rodriguez, J. Leskovec, B. Schölkopf, Structure and dynamics of information pathways in online media, in: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, 2013, pp. 23–32.
[21] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, 1979.
[22] M.E. Gaston, M. desJardins, Agent-organized networks for dynamic team formation, in: Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS-2005, 2005, pp. 230–237.
[23] O. Goldschmidt, D.S. Hochbaum, A polynomial algorithm for the k-cut problem for fixed k, Math. Oper. Res. 19 (1) (1994) 24–37.
[24] M. Hoefer, D. Vaz, L. Wagner, Hedonic coalition formation in networks, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI-2015, 2015, pp. 929–935.
[25] R.M. Karp, Reducibility Among Combinatorial Problems, Springer, 1972.
[26] J. Kleinberg, É. Tardos, Balanced outcomes in social exchange networks, in: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, 2008, pp. 295–304.
[27] J. Könemann, K. Larson, D. Steiner, Network bargaining: using approximate blocking sets to stabilize unstable instances, Theory Comput. Syst. 57 (3) (2015) 655–672.
[28] C.E. Leiserson, R.L. Rivest, C. Stein, T.H. Cormen, Introduction to Algorithms, The MIT Press, 2001.
[29] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, Z. Ghahramani, Kronecker graphs: an approach to modeling networks, J. Mach. Learn. Res. 11 (2010) 985–1042.
[30] S.T. McCormick, M.R. Rao, G. Rinaldi, Easy and difficult objective functions for max cut, Math. Program. 94 (2–3) (2003) 459–466.
[31] R. Meir, J.S. Rosenschein, E. Malizia, Subsidies, stability, and restricted cooperation in coalitional games, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI-2011, 2011, pp. 301–306.
[32] I. Milchtaich, E. Winter, Stability and segregation in group formation, Games Econ. Behav. 38 (2) (2002) 318–346.
[33] M.G. Rodriguez, D. Balduzzi, B. Schölkopf, Uncovering the temporal dynamics of diffusion networks, preprint, arXiv:1105.0697, 2011.
[34] W. Saad, Z. Han, T. Basar, M. Debbah, A. Hjorungnes, A selfish approach to coalition formation among unmanned air vehicles in wireless networks, in: Proceedings of the 1st International Conference on Game Theory for Networks, GameNets-2009, 2009, pp. 259–267.
[35] T. Sandholm, K. Larson, M. Andersson, O. Shehory, F. Tohmé, Coalition structure generation with worst case guarantees, Artif. Intell. 111 (1–2) (1999) 209–238.
[36] L. Sless, N. Hazon, S. Kraus, M. Wooldridge, Forming coalitions and facilitating relationships for completing tasks in social networks, in: Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS-2014, 2014, pp. 261–268.
[37] I. Stanton, G. Kliot, Streaming graph partitioning for large distributed graphs, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2012, pp. 1222–1230.
[38] S.C. Sung, D. Dimitrov, On core membership testing for hedonic coalition formation games, Oper. Res. Lett. 35 (2) (2007) 155–158.
[39] T. Voice, M. Polukarov, N.R. Jennings, Coalition structure generation over graphs, J. Artif. Intell. Res. 45 (1) (2012) 165–196.