Socially Aware Coalition Formation with Bounded Coalition Size

Chaya Levinger, Amos Azaria and Noam Hazon

Abstract

In many situations when people are assigned to coalitions the assignment must be social aware, i.e, the utility of each person depends on the friends in her coalition. Additionally, in many situations the size of each coalition should be bounded. This paper initiates the study of such coalition formation scenarios in both weighted and unweighted settings. We show that finding a partition that maximizes the utilitarian social welfare is computationally hard, and provide a polynomial-time approximation algorithm. We also investigate the existence and the complexity of finding stable partitions. Namely, we show that the Contractual Strict Core (CSC) is never empty, but the Strict Core (SC) of some games is empty. Finding partitions that are in the CSC is computationally easy, but finding partitions that are in the SC is hard. The analysis of the core is more involved. For the weighted setting, the core may be empty. We thus concentrate on the unweighted setting. We show that when the coalition size is bounded by 3 the core is never empty, and we present a polynomial time algorithm for finding a member of the core. When the coalition size is greater, we provide additive and multiplicative approximations of the core. In addition, we show in simulation over 100 million games that a simple heuristic always finds a partition that is in the core.

1 Introduction

Suppose that a group of travelers who are located at some origin, would like to reach the same destination, and later return. Each of the travelers has her own vehicle; but each traveler has a preference related to whom will be with her in the vehicle. Namely, each traveler would rather share a vehicle with as many as possible of her friends during the ride, and thus the utility of each traveler is the number of friends traveling with her. However, the vehicles have a limited capacity; this capacity can either be a physical constraint of the vehicles, or the maximal number of travelers willing to travel together. How should the travelers be assigned to vehicles in order to maximize the social welfare (the sum of all travelers' utilities)? Can the travelers be organized such that no subgroup of travelers will want to leave their current group and join together? Similar questions raise when assigning students to dormitories, colleagues to office-rooms and workers to project teams. In these settings, it might be that the utility of each person does not depend only on the number of friends, but also on the intensity of friendship.

This set of problems falls within hedonic games [13], in which a set of agents are partitioned into coalitions, and the utility for each agent depends only on the coalition that she is a member of. Additively Separable Hedonic Games (ASHGs) [9] are a special type of hedonic games, in which each agent has a value for any other agent, and the utility she assigns to a coalition is the sum of the values she assigns to its members. In ASHGs there is usually no restriction on the number of agents that are allowed to belong to a coalition. However, in our group of travelers example, the vehicles have physical capacity, and thus, there is an upper bound on the size of each coalition. Despite this restriction being natural, it is scarcely studied in the domain of hedonic games.

In this paper, we initiate the study of hedonic games with bounded coalition size. Specifically, we concentrate on symmetric ASHGs, in which the value an agent assigns to another agent is non-negative and it is equal to the value that the other agent assigns to her; we refer to these settings as the weighted settings. We also study simple symmetric ASHGs, in which the value an agent assigns to other agents is either 0 or 1; we refer to these settings as the unweighted settings. These models capture many situations, such as social and friendship relations. We begin by studying the problem

of finding a partition that maximizes the utilitarian social welfare. We show that this problem is computationally hard for any coalition size bound $k > 2$, even in the unweighted setting. Therefore, we provide a polynomial-time approximation algorithm. We prove that in the unweighted setting, the algorithm has an approximation ratio of $\frac{1}{k-1}$, and in the weighted setting it has an approximation ratio of $\frac{1}{k}$.

We then study stability aspects of the problem. That is, we investigate the existence and the complexity of finding stable partitions. Namely, we show that the Contractual Strict Core (CSC) is never empty, but the Strict Core (SC) of some games is empty. Finding partitions that are in the CSC is computationally easy, but finding partitions that are in the SC is hard. The analysis of the core is more involved. For the weighted setting, the core may be empty; we provide an example for $k = 3$. We thus concentrate on the unweighted setting. We show that for k=3 the core is never empty, and we present a polynomial time algorithm for finding a member of the core. For $k > 3$ it is unclear whether the core can be empty, and how to find a partition in the core. Therefore, we investigate additive and multiplicative approximations of the core. In addition, we show in simulation over 100 million games that a simple heuristic always finds a partition that is in the core.

To summarize, the contribution of this work is being the first systematic study of additively separable hedonic games with bounded coalition size. Namely, we provide an approximation algorithm for maximizing the utilitarian social welfare and study the computational aspects of several stability concepts.

2 Related Work

Dreze and Greenberg [1980] initiated the study of hedonic games, in which the utility for each agent depends only on the coalition that she is a member of. Stability concepts of hedonic games were further analyzed in [6] and [10]. For more details, see the survey of Aziz et al. [2016]. A special case is Additively Separable Hedonic Games (ASHGs) [9], in which each agent has a value for any other agent, and the utility she assigns to a coalition is the sum of the values she assigns to its members. The computational aspects of ASHGs are analyzed in [12, 5, 21, 24, 1, 4, 7]. None of these works imposed any restriction on the size of the coalitions.

Indeed, there are few papers that impose a restriction on the size of the coalitions. Wright and Vorobeychik [2015] study a model of ASHG where there is an upper bound on the size of each coalition. Within their model, they propose a strategyproof mechanism that achieves good and fair experimental performance, despite not having a theoretical guarantee. Flammini et al. [2021] study the online partition problem. Similar to our work, they also consider the scenario that the coalitions are bounded by some number. They consider two cases for the value of a coalition, the sum of the weights of its edges, which is similar to our work, and the sum of the weights of its edges divided by its size. However, in both cases they only consider the online version, i.e., the agents arrive sequentially and must be assigned to a coalition as they arrive. This assignment cannot be adapted later on, and must remain. They show that a simple greedy algorithm achieves an approximation ratio of $\frac{1}{k}$ when the value of the coalition is the sum of the weights. Cseh et al. [2019] require the partition to be composed of exactly k coalitions, and also assume a predefined set of size constraints. Each coalition is required to exactly match its predefined size. They study the complexity of finding a Pareto optimal partition, as well as the complexity of deciding whether a given partition is Pareto optimal. Bilò et al. [2022] consider the same settings as Cseh et al. Since classical stability notions are infeasible in their setting, they study three different types of swap stability, and analyze the existence, complexity, and efficiency of stable outcomes. Note that almost all other works analyzing ASHGs assume that an agent may assign a negative value to another agent. Otherwise, since they do not impose any restrictions on the coalition size, the game becomes trivial, as the grand coalition is always an optimal solution. We found two exceptions who restrict the value each agent assigns to other agents to be either 0 or 1. Namely, Sless et al. [2018] study the setting in which the agents

must be partitioned into exactly k coalitions, without any restriction on each coalition's size. Li et al. [2023] study the setting in which the agents must be partitioned into exactly k coalitions that are almost equal in their size.

In our study of the core, we consider additive and multiplicative approximations for the core. We note that similar approximations were discussed in other settings [22, 20, 15]. Recent work by Li et al. [2023] has considered both additive and multiplicative approximations of the core in ASHGs.

3 Preliminaries

Let $V = \{v_1, ..., v_n\}$ be a set of agents, and let $G(V, E)$ be a weighted undirected graph representing the social relations between the agents. For every edge $e \in E$, the weight of the edge, $w(e)$, is positive. In the unweighted setting, all weights are set to 1. A k-bounded coalition is a coalition of size at most k. A k-bounded partition P is a partition of the agents into disjoint k-bounded coalitions. Given a coalition $S \in P$, and $v \in S$, let $N(v, S)$ be the set of immediate neighbors of $v \in V$ in S, i.e., $N(v, S) = \{u \in S : (v, u) \in E\}$. Let $W(v, S)$ be the sum of weights of immediate neighbors of $v \in V$ in S, i.e., $W(v, S) = \sum_{u \in N(v,S)} w((v, u))$. Note that in the unweighted setting, $W(v, S) = |N(v, S)|$. An *additively separable hedonic game with bounded coalition size* is a tuple (G, k), where for every k-bounded partition P, coalition $S \in P$, and $v \in S$, the agent v gets utility $W(v, S)$. We denote the utility of v given a k-bounded partition P, by $u(v, P)$. Given a tuple (G, k), the goal is to find a k-bounded partition P that satisfies efficiency or stability properties.

We consider the following efficiency or stability concepts:

- The utilitarian social welfare of a partition P, denoted $u(P)$, is the sum of the utilities of the agents. That is, $u(P) = \sum_{v \in V} u(v, P)$. A *MaxUtil* k-bounded partition P is a partition with maximum $u(P)$.
- A k-bounded coalition S is said to *strongly block* a k-bounded partition P if for every $v \in S$, $W(v, S) > u(v, P)$. A k-bounded partition P is in the *Core* if it does not have any strongly blocking k-bounded coalitions.
- A k-bounded coalition S is said to *weakly block* a k-bounded partition P if for every $v \in S$, $W(v, S) \geq u(v, P)$, and there exists some $v \in S$ such that $W(v, S) > u(v, P)$. A k-bounded partition P is in the *Strict Core (SC)* if it does not have any weakly blocking k-bounded coalitions.
- Given a partition P and a set S, let P^{-S} be the partition when S breaks off. That is, $P^{-S} = \{S\} \cup \bigcup_{C \in P} \{C \setminus S\}$. A k-bounded partition P is in the *Contractual Strict Core (CSC)* if for any weakly blocking k-bounded coalition S, there exists at least one agent v such that $u(v, P^{-S}) < u(v, P)$.

Due to space constraints, some proofs are deferred to the appendix.

4 Efficiency

We begin with the elementary concept of efficiency, which is to maximize the utilitarian social welfare.

Definition 4.1 (MaxUtil problem)**.** *Given a coalition size limit k and a graph G, find a MaxUtil k-bounded partition.*

(a) The graph G

(b) An optimal partition in G

Figure 1: An example for the MaxUtil problem where $k = 3$.

For example, given the unweighted graph in Figure 1a and a coalition size limit $k = 3$, let $P = \{\{v_1, v_3, v_6\}, \{v_2, v_4, v_7\}, \{v_5, v_8\}\}$, shown in Figure 1b. The utilitarian social welfare of this partition, $u(P)$ equals 14. Indeed, this is an optimal 3-bounded partition, since there is no other 3-bounded partition with higher social welfare. Clearly, the decision variant of the MaxUtil problem is to decide whether there exists a k-bounded partition with a utilitarian social welfare of at least υ.

4.1 The Hardness of the MaxUtil Problem

The MaxUtil problem when $k = 2$ is equivalent to the maximum (weight) matching problem, and thus it can be computed in polynomial time [14]. However, our problem becomes intractable when $k \geq 3$ even in the unweighted setting. For the hardness proof, we define for each $k \in \mathbb{N}$ the $Cliques_k$ problem, which is as follows.

Definition 4.2 ($Cliques_k$)**.** *Given an undirected and unweighted graph $G(V, E)$, decide whether V can be partitioned into disjoint cliques, such that each clique is composed of exactly k vertices.*

Clearly, $Cliques_2$ can be decided in polynomial time by computing a maximum matching of the graph G, M, and testing whether $|M| = \frac{|V|}{2}$. However, $Cliques_k$ becomes hard when $k \geq 3$.

Lemma 1. *$Cliques_k$ is NP-Complete for every $k \geq 3$.*

Theorem 2. *The decision variant of the MaxUtil problem is NP-Complete, in the unweighted and weighted settings.*

4.2 Approximation of the MaxUtil Problem

Since we showed that the MaxUtil problem is NP-Complete, we now provide the Match and Merge (MnM) algorithm (Algorithm 1), which is a polynomial-time approximation algorithm for any $k \geq 3$. The algorithm consists of $k - 1$ rounds. Each round is composed of a matching phase followed by a merging phase. Specifically, in round l MnM computes a maximum (weight) matching, $M_l \subseteq E_l$, for G_l (where $G_1 = G$). In the merging phase, MnM creates a graph G_{l+1} that includes a unified node for each pair of matched nodes. G_{l+1} also includes all unmatched nodes, along with their edges to the unified nodes (lines 10-13). Clearly, each node in V_l is composed of up-to l nodes from V_1. Finally, MnM returns the k-bounded partition, P, of all the matched sets. For example, given the graph G_1 in Figure 2a and $k = 4$, the algorithm finds a maximum matching $M_1 = \{(v_1, v_2), (v_3, v_4)\}$ shown in Figure 2b. It then creates the graph G_2, as shown in Figure 2c, and finds a maximum matching for it, $M_2 = \{(v_{3,4}, v_5)\}$ shown in Figure 2d. It then creates the graph G_3, as shown in Figure 2e, and finds a maximum matching for it, $M_3 = \{(v_{3,4,5}, v_6)\}$. Finally, MnM created the graph G_4, as shown in Figure 2f, and returns the 4-bounded partition $P = \{v_1, v_2\}, \{v_3, v_4, v_5, v_6\}$. We note that by the algorithm construction, a unified node $v_{i_1,\ldots,i_l}$, is created by merging nodes v_{i_1} and v_{i_2}, and then by merging v_{i_1,i_2} and v_{i_3}, and so on.

Algorithm 1: Match and Merge (MnM)

Input: A graph $G(V, E)$ and a limit k
Result: A k-bounded partition P of V.
$G_1(V_1, E_1) \leftarrow G(V, E)$
for $l \leftarrow 1$ *to* $k-1$ **do**
  $M_l \leftarrow$ maximum (weight) matching in G_l
  $G_{l+1} = (V_{l+1}, E_{l+1}) \leftarrow$ an empty graph
  $V_{l+1} \leftarrow V_l$
  for every $(v_{i_1,...,i_l}, v_j) \in M_l$ **do**
    add vertex $v_{i_1,...,i_l,j}$ to V_{l+1}
    remove $v_{i_1,...,i_l}, v_j$ from V_{l+1}
  for every $v_{i_1,...,i_{l+1}} \in V_{l+1}$ **do**
    for every $v_q \in V_{l+1}$ **do**
      if $(v_{i_1,...,i_l}, v_q) \in E_l$ or $(v_{i_{l+1}}, v_q) \in E_l$ **then**
        add $(v_{i_1,...,i_{l+1}}, v_q)$ to E_{l+1}
$P \leftarrow$ an empty partition
for every $v_{i_1,...,i_j} \in G_k$ **do**
  add the set $\{v_{i_1}, ..., v_{i_j}\}$ to P
return P

4.3 Approximation Ratio for Unweighted Setting

We first show that MnM provides an approximation ratio of $\frac{1}{k-1}$ for the MaxUtil problem in the unweighted setting. For that end, we first prove the following lemma related to the possible edges in every G_l, $l > 1$.

Lemma 3. *Given $\hat{v} = v_{i_1,...,i_l} \in V_l$, if there exist $v_i, v_j \in V_l$, $v_i \neq v_j$ such that $(v_i, v_{i_n}), (v_j, v_{i_m}) \in E$ for some $1 \leq n \leq m \leq l$, then $n = m$.*

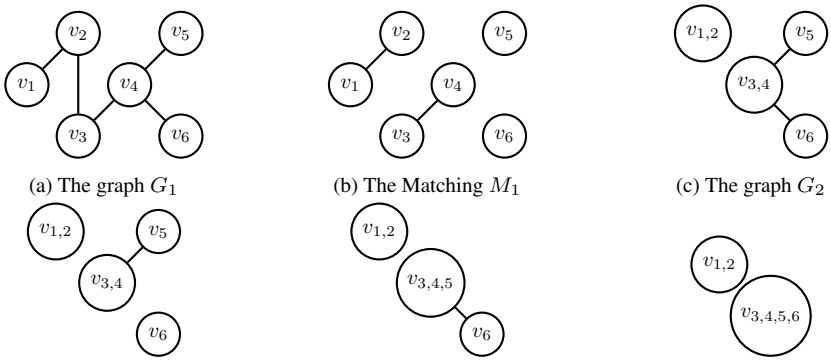
Proof. Observe that for every $v_i, v_j \in V_l$ where $l > 1$, $(v_i, v_j) \notin E$, since M_1 is a maximum matching in G_1. Assume by contradiction and without loss of generality that $n < m$. If $n = 1$ and $m = 2$, then the path $v_i \rightarrow v_{i_n} \rightarrow v_{i_m} \rightarrow v_j$ is an M_1-augmenting path in G_1 ([14]), contrary to the fact that M_1 is a maximum matching in G_1. Therefore, $m \geq 3$. Consider the graph G_2, since $v_{i_m}, v_j \in V_2$, there exists an edge $(v_j, v_{i_m}) \in E$, contrary to our observation.

□

We now present a hypothetical procedure (Procedure 2) that is provided with a solution to the MaxUtil problem, which is a k-bounded partition (of G) Opt, a graph G_l (as defined in Algorithm 1), and a corresponding round index l. Without loss of generality, we assume that every set $S \in Opt$ is a connected component. Let $O = \{v_o | \{v_o\} \in Opt \text{ and } v_o \in V_2\}$. That is, $|O|$ is the number of singletons in the partition Opt that are also not matched in M_1. We show that Procedure 2 finds a matching, and we provide a lower bound on the size of this matching (the number of edges in it). We further show that MnM is guaranteed to perform at least as well as this procedure, which, as we show, results in an approximation ratio of $\frac{1}{k-1}$ for every $k \geq 3$.

Lemma 4. *Procedure 2 finds a matching, R_l, in the graph G_l, such that $|R_l| \geq (|V| - 2|M_1| - \sum_{i=2}^{l-1} |M_i| - |O|)/(k-1)$, where $l > 1$.*

Proof. We first show that Procedure 2 finds a matching, R_l, in the graph G_l. At each iteration of the loop in line 8, we add an edge between a single node, v_q, and a unified node, $v_{i_1,...,i_l}$. We consider



(a) The graph G_1 (b) The Matching M_1 (c) The graph G_2

(d) The Matching M_2 (e) The graph G_3 (f) The graph G_4

Figure 2: An example for Algorithm 1 where $k = 4$.

Procedure 2: Find matching

1 **Input**:
2 A k-bounded partition (of G) Opt
3 A graph $G_l = (V_l, E_l)$
4 **Result**:
5 A matching in G_l $R_l \leftarrow$ an empty matching
6 **for** each $v_i \in V_l$ such that $\{v_i\} \in Opt$ **do**
7 remove v_i from V_l
8 **for** each $v_q \in V_l$ **do**
9 let $\hat{v}$ be a vertex $v_{i_1,...,i_l}$ such that $(v_q, \hat{v}) \in E_l$ and for some $1 \leq j \leq l$, v_q and v_{i_j} belong to the same set in Opt
10 **for** each $v_n \neq v_q$ **do**
11 **if** $(v_n, \hat{v}) \in E_l$ and exists $1 \leq m \leq l$, s.t. v_{i_m} and v_n belong to the same set in Opt **then**
12 remove v_n from V_l
13 add $(v_q, \hat{v})$ to R_l
14 **return** R_l

each single node only once. Therefore, it is not possible to add a single node twice to R_l. Similarly, each time a unified node is added to R_l, every single node $v_n \neq v_q$ such that v_{i_m} and v_n belong to the same set in Opt, for some $1 \leq m \leq l$, is removed from V_l. Therefore, a unified node is not added more than once. That is, R_l is a matching in G_l.

We now show a lower bound on the size of $|R_l|$. Let $V_l' = \{v_i | v_i \in V_l\}$, i.e., the set of all the single nodes in G_l. In line 12 we remove nodes only when $m = j$ (according to Lemma 3). Given $\hat{v} = v_{i_1,...,i_l}$, there are at most $k-1$ different nodes, $v_{j_1}, ..., v_{j_{k-1}}$ that are in the same set with $\hat{v}$ in Opt. Therefore, in each iteration of the loop in line 8, we remove at most $k-2$ single nodes in line 12 while adding one edge to R_l in line 13. Thus, at least $\frac{1}{k-1}$ of the single nodes in V_l (who are not in O) are matched to a unified node. Therefore, $|R_l| \geq \frac{|V_l'|-|O|}{k-1}$. Now, $|V_2'| = |V_1| - 2|M_1|$. In addition, at each iteration $l > i > 1$, $|M_i|$ single nodes are each added to a unified node. Therefore, $|V_l'| = |V_1| - 2|M_1| - \sum_{i=2}^{l-1} |M_i|$. In addition, $V = V_1$. Overall, $|R_l| \geq (|V| - 2|M_1| - \sum_{i=2}^{l-1} |M_i| - |O|)/(k-1)$. □

Theorem 5. *Algorithm 1 provides a solution for the MaxUtil problem with an approximation ratio of $\frac{1}{k-1}$ for every $k \geq 3$, in the unweighted setting.*

Proof. Let P be the k-bounded partition returned by Algorithm 1. Clearly, $u(P) \geq 2 \cdot \sum_{i=1}^{k-1} |M_i|$. For every $l \geq 1$, M_l is a maximum matching and thus $|M_l| \geq |R_l|$. In addition, according to Lemma 4, $|R_l| \geq \frac{|V|-2|M_1|-\sum_{i=2}^{l-1}|M_i|-|O|}{k-1}$.

Therefore,

$$u(P) \geq 2 \cdot \sum_{i=1}^{k-1} |M_i| = 2|M_1| + 2 \cdot \sum_{i=2}^{k-1} |M_i|.$$

$$\sum_{i=2}^{k-1} |M_i| = |M_2| + |M_3| + ... + |M_{k-1}| \geq$$

$$|M_2|+|M_3|+...+|M_{k-2}|+|R_{k-1}| \geq |M_2|+|M_3|+...+|M_{k-2}|+\frac{|V|-|O|-2|M_1|-|M_2|-...-|M_{k-2}|}{k-1} =$$

$$\frac{|V|-|O|-2|M_1|}{k-1}+\frac{k-2}{k-1}\sum_{i=2}^{k-2}|M_i| \geq (1+\frac{k-2}{k-1})\cdot\frac{|V|-|O|-2|M_1|}{k-1}+(\frac{k-2}{k-1})^2\sum_{i=2}^{k-3}|M_i| \geq ... \geq$$

$$(1+\frac{k-2}{k-1}+(\frac{k-2}{k-1})^2+...+(\frac{k-2}{k-1})^{k-3})\cdot\frac{|V|-|O|-2|M_1|}{k-1}+(\frac{k-2}{k-1})^{k-2}\sum_{i=2}^{k-1-(k-2)}|M_i| =$$

$$\sum_{i=0}^{k-3}\Big((\frac{k-2}{k-1})^i\cdot\frac{|V|-|O|-2|M_1|}{k-1}\Big).$$

That is,

$$u(P) \geq 2|M_1| + 2 \cdot \sum_{i=0}^{k-3}\Big((\frac{k-2}{k-1})^i\cdot\frac{|V|-|O|-2|M_1|}{k-1}\Big) =$$

$$2|M_1| + 2 \cdot \frac{|V|-|O|-2|M_1|}{k-1} \cdot \frac{(\frac{k-2}{k-1})^{(k-2)}-1}{\frac{k-2}{k-1}-1} =$$

$$2|M_1| + 2(|V|-|O|-2|M_1|) \cdot \frac{(\frac{k-2}{k-1})^{(k-2)}-1}{(k-1)(\frac{k-2}{k-1}-1)} =$$

$$2|M_1| - 2(|V|-|O|-2|M_1|)((\frac{k-2}{k-1})^{(k-2)}-1) =$$

$$2(|V|-|O|)(1-(\frac{k-2}{k-1})^{(k-2)}) - 2|M_1|(1-2\cdot(\frac{k-2}{k-1})^{(k-2)}).$$

Next, we show that

$$(1-2\cdot(\frac{k-2}{k-1})^{(k-2)}) \geq 0.$$

Let

$$f(k) = (\frac{k-2}{k-1})^{k-2}, \text{ for } k \geq 3.$$

Thus

$$f'(k) = \frac{(k-2)^{k-2}\big(ln(\frac{k-2}{k-1})(k-1)+1\big)}{(k-1)^{k-1}}.$$

Now, $\frac{(k-2)^{k-2}}{(k-1)^{k-1}} > 0$. In addition, it is known that $ln(x) \leq x - 1$ [19], and thus

$$ln(\frac{k-2}{k-1})(k-1) + 1 \leq -\frac{1}{k-1}(k-1) + 1 = 0$$

Therefore, for all $k \geq 3$, $f'(k) \leq 0$ and $f(k) \leq f(3) = \frac{1}{2}$.

Overall, since $|M_1| \leq \frac{|V|-|O|}{2}$,

$$u(P) \geq 2(|V| - |O|)(1 - (\frac{k-2}{k-1})^{(k-2)}) - 2 \cdot \frac{|V| - |O|}{2}(1 - 2 \cdot (\frac{k-2}{k-1})^{(k-2)}) = |V| - |O|.$$

Now, since in Opt there are at least $|O|$ singletons, then $u(Opt)$ is at most $(|V| - |O|) \cdot (k-1)$, which occurs when all nodes are partitioned into cliques of size k (except those in O). That is,

$$u(P) \geq \frac{u(Opt)}{k-1}.$$

□

Since finding a maximum matching in a graph can be computed in $O(|E|\sqrt{|V|})$, Algorithm 1 runs in $O(kn^{2.5})$ time.

4.4 Approximation Ratio for Weighted Setting

We now show that MnM provides an approximation ratio of $\frac{1}{k}$ for the MaxUtil problem in the weighted setting. Specifically, we show that the first step of the algorithm, which finds a maximum weight matching, provides an approximation ratio of $\frac{1}{k}$.

Theorem 6. *Algorithm 1 provides a solution for the MaxUtil problem with an approximation ratio of $\frac{1}{k}$ for every $k \geq 3$, in the weighted setting.*

5 Stability

When considering a stability concept c, we analyze the following two problems:

- Existence: determine whether for any (G, k) there exists a partition that satisfies c.
- Finding: given (G, k), decide if there exists a partition that satisfies c and if so, find such a partition.

5.1 Core

We begin by showing that the core in the weighted setting may be empty. Specifically, Figure 3 provides an example of such a graph for $k = 3$. We use a computer program that iterates over all possible 3-bounded partitions, for verifying that each such 3-bounded partition has a strongly blocking 3-bounded coalition.

We thus concentrate on the unweighted setting. First, we show that for $k = 3$ the core is never empty. We present Algorithm 3, a polynomial time algorithm that finds a 3-bounded partition P in the core. The algorithm begins with all agents in singletons and iteratively considers for each 3-bounded coalition whether it strongly blocks the current partition.

Theorem 7. *In the unweighted setting, there always exists a 3-bounded partition in the core, and it can be found in polynomial time.*

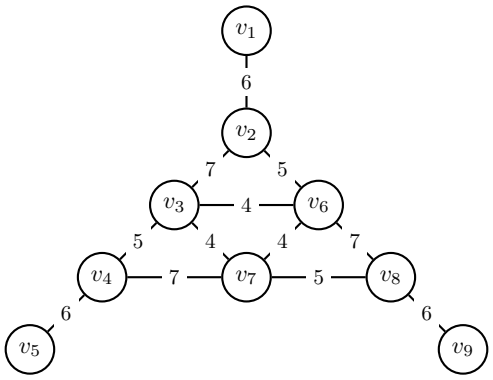


Figure 3: An example of a weighted graph in which the core is empty, for $k = 3$.

Algorithm 3: Finding a 3-bounded partition in the core

```
1  Input: A graph G(V, E)
   Result: A 3-bounded partition P of V in the core.
2  P ← {{v} for every v ∈ V}
3  V' ← V
4  outerLoop:
5  for S ⊂ V', such that |S| = 2 OR |S| = 3 do
6      if ∀v ∈ S, W(v, S) > u(v, P) then
7          P ← P^{-S}
8          if S is clique of size 3 then
9              V' ← V' \ S
10         goto outerLoop
11 return P
```

Proof. Consider Algorithm 3. Note that for every 3-bounded partition P, if $S \in P$ is a clique of size 3 then every $v \in S$ cannot belong to any strongly blocking coalition. Therefore, Algorithm 3 removes such vertices from V' (in line 9). Clearly, if Algorithm 3 terminates, the 3-bounded partition P is in the core. We now show that Algorithm 3 must always terminate, and it runs in polynomial time. The algorithm initiates a new iteration (line 4) whenever the if statement in line 6 is true, which can happen when the blocking coalition S, is one of the following:

- Only singletons (i.e., two or three singletons). Then, $u(P)$ increases by at least 2.
- One agent from a coalition in which she has one neighbor, and two singleton agents. Then, $u(P)$ increases by at least 2.
- Two agents, each from a coalition with a single neighbor, and one singleton agent. Then, S must be a clique of size 3, which increases $u(P)$ by 2.
- Three agents, each from a coalition with a single neighbor. Then, S must also be a clique of size 3; however, $u(P)$ remains the same.

Overall, either $u(P)$ has increased by at least 2 or S is a clique of size 3 and thus its vertices are removed from further consideration (in line 9). Since $u(P)$ is bounded by $2|E|$ and the number of vertices is finite, the algorithm must terminate after at most $|E| + |V|/3$ iterations. □

For $k > 3$ it is unclear whether the core can be empty, and how to find a partition in the core. Therefore, we now investigate additive and multiplicative approximations of the core, which are defined as follows.

Definition 5.1 (Additive approximation)**.** *A k-bounded coalition S is said to* ϵ_a-strongly block *a k-bounded partition P if it improves the utility of each of its members by more than an additive factor of ϵ_a. That is, for every $v \in S$, $W(v,S) > u(v,P) + \epsilon_a$. A k-bounded partition P is in the* ϵ_a-core *if it does not have any ϵ_a-strongly blocking k-bounded coalitions.*

The *ϵ_m-core*, which is the multiplicative approximation of the core is defined similarly. That is, a k-bounded coalition S is said to *ϵ_m-strongly block* a k-bounded partition P if for every $v \in S$, $W(v,S) > \epsilon_m \cdot u(v,P)$.

We now show that for $k > 3$ the ϵ_a-core, for $\epsilon_a = \lfloor \frac{k}{2} \rfloor - 1$, is never empty. We present Algorithm 4, a polynomial time algorithm that finds a k-bounded partition P in the ϵ_a-core. The algorithm begins with all agents in singletons and iteratively considers for each k-bounded coalition whether it ϵ_a-strongly blocks the current partition.

Algorithm 4: Finding a k-bounded partition in the ϵ_a-core

```
1  Input: A graph G(V, E)
2  A limit k
3  An additive factor ε_a
   Result: A k-bounded partition P of V in the ε_a-core.
4  P ← {{v} for every v ∈ V}
5  V' ← V
6  outerLoop:
7  for S ⊂ V', such that 1 < |S| ≤ k do
8      if ∀v ∈ S, W(v,S) > u(v,P) + ε_a then
9          P ← P^{-S}
10         if ∀v ∈ S, W(v,S) ≥ k − 1 − ε_a then
11             V' ← V' \ S
12         goto outerLoop
13 return P
```

Theorem 8. *For $\epsilon_a = \lfloor \frac{k}{2} \rfloor - 1$, there always exists a k-bounded partition in the ϵ_a-core, and it can be found in polynomial time.*

Proof. Consider Algorithm 4. Note that for every k-bounded partition P, and $S \in P$, if for every $v \in S, W(v,S) \geq k-1-\epsilon_a$ then every $v \in S$ cannot belong to any ϵ_a-strongly blocking k-bounded coalition. Therefore, Algorithm 4 removes such vertices from V' (in line 11). Clearly, if Algorithm 4 terminates, the k-bounded partition P is in the ϵ_a-core. We now show that for $\epsilon_a = \lfloor \frac{k}{2} \rfloor - 1$, Algorithm 4 must always terminate, and it runs in polynomial time.

First, we show that $u(P)$ never decreases. The algorithm initiates a new iteration (line 6) whenever the if statement in line 8 is true. This can happen only if for every v in the blocking coalition S, $u(v,P)$ is less than $k - 1 - \lfloor \frac{k}{2} \rfloor + 1$ (since $W(v,S)$ is at most $k-1$). Therefore, $u(v,P) < k - \frac{k}{2} + \frac{1}{2} = \frac{k}{2} + \frac{1}{2}$. Since $u(v,P)$ is a natural number, $u(v,P) \leq \frac{k}{2} - \frac{1}{2}$. When S breaks off, the social welfare decreases by at most $2 \cdot \sum_{v \in S} u(v,P)$, and increases by at least $\sum_{v \in S} (u(v,P) + \lfloor \frac{k}{2} \rfloor)$. Since $\sum_{v \in S} (u(v,P) + \lfloor \frac{k}{2} \rfloor) \geq \sum_{v \in S} (u(v,P) + \frac{k}{2} - \frac{1}{2}) \geq \sum_{v \in S} (u(v,P) + u(v,P)) = 2 \cdot \sum_{v \in S} u(v,P)$, then $u(P)$ never decreases.

Next, we show that if $u(P)$ remains the same, then vertices are removed from further consideration (in line 11). Observe that $u(P)$ remaining the same entails that $2 \cdot \sum_{v \in S} u(v,P) = \sum_{v \in S} (u(v,P) + \lfloor \frac{k}{2} \rfloor)$. That is, $\sum_{v \in S} u(v,P) = \sum_{v \in S} \lfloor \frac{k}{2} \rfloor$. Recall that for every $v \in S$, $u(v,P) \leq \frac{k}{2} - \frac{1}{2}$ and note that $\frac{k}{2} - \frac{1}{2} \leq \lfloor \frac{k}{2} \rfloor$. Therefore, if $u(P)$ remains the same, then for every $v \in S$, $u(v,P) = \frac{k}{2} - \frac{1}{2}$. Now, since for every $v \in S$, $W(v,S) > u(v,P) + \lfloor \frac{k}{2} \rfloor - 1$ and $\epsilon_a \geq 1$, then $W(v,S) > \frac{k}{2} - \frac{1}{2} + \frac{k}{2} - \frac{1}{2} - 1 = k - 2 \geq k - 1 - \epsilon_a$. Therefore, all $v \in S$ are removed from further consideration.

Overall, either $u(P)$ has increased by at least 2 or vertices are removed from further consideration. Since $u(P)$ is bounded by $2|E|$ and the number of vertices is finite, the algorithm must terminate after at most $|E| + \frac{|V|}{k}$ iterations. □

Next, we show that for $k > 3$ the ϵ_m-core, for $\epsilon_m = 2$, is never empty. We use Algorithm 4, with the following changes:

- The input of the algorithm is ϵ_m instead of ϵ_a (in line 3).
- In line 8, we check if for every v in S, $W(v,S) > \frac{u(v,P)}{\epsilon_m}$.
- In line 10, we check if for every v in S, $W(v,S) \geq \frac{k-1}{\epsilon_m}$.

We show that this algorithm finds a k-bounded partition P in the ϵ_m-core in polynomial time.

Theorem 9. *For $\epsilon_m = 2$, there always exists a k-bounded partition in the ϵ_m-core, and it can be found in polynomial time.*

Finally, we show in simulation that a simple heuristic always finds a partition that is in the core. Our heuristic function works as follows:

1. Start with a k-bounded partition P, where all the agents are singletons.
2. Iterate randomly over all the k-bounded coalitions until a coalition S is found, which strongly blocks the partition P.
3. Update P to be P^{-S}, and return to step (2).

The heuristic terminates when either there is no strongly blocking coalition for the partition P (i.e., P is in the core), or when in 100 consecutive iterations all of the partitions have already been seen. In the later case, we restart the heuristic.

We test our heuristic function for $k = 5$ over more than 100 million random graphs of different types:

- Random graphs of size 30 with probability of 0.5 for rewiring each edge.
- Random trees of size 30.
- Random connected Watts–Strogatz small-world graphs of size 30, where each node is joined with its 5 nearest neighbors in a ring topology and with a probability of 0.5 for rewiring each edge.

Our heuristic always found a k-bounded partition that is in the core. Moreover, we had to restart the heuristic in only 33 instances, and then a k-bounded partition in the core was found.

5.2 Strict Core (SC)

We first show that for every size limit, k, there is at least one graph where there is no k-bounded partition in the strict core. Indeed, given a size limit k, we build the graph $G(V, E)$, which is a clique of size $k + 1$. For every partition P of V, let S be a coalition in P such that $|S| < k$. Now, any set of agents of size k that also contains some $v \in S$ is a weakly blocking k-bounded coalition for P. Furthermore, even verifying the existence of the strict core is a hard problem.

Definition 5.2 (SC existence problem)**.** *Given a coalition size limit k and a graph G, decide whether a k-bounded partition exists that is in the strict core.*

Theorem 10. *The SC existence problem is NP-hard.*

5.3 Contractual Strict Core (CSC)

We show that the CSC is never empty. Indeed, given any (G, k), the following algorithm finds a k-bounded partition in the CSC:

1. Start with a k-bounded partition P, where all the agents are singletons.
2. Iterate over all the coalitions in P until two coalitions, S_1, S_2, are found, such that $|S_1| + |S_2| \leq k$ and $u(P) < u(P^{-S_1 \cup S_2})$.
3. Update P to be $P^{-S_1 \cup S_2}$, and return to step (2).

The algorithm terminates when step 2 does not find two coalitions that meet the required conditions.

Theorem 11. *There always exists a k-bounded partition in the CSC, and it can be found in polynomial time.*

Proof. At each iteration, the number of the coalitions in P decreases and thus the algorithm must terminate after at most $k - 1$ iterations. Consider the k-bounded partition P when the algorithm terminates. Clearly, there are no two coalitions in P that can benefit from breaking off and joining together. In addition, observe that every coalition $S \in P$ is a connected component. Thus, no coalition $S' \subsetneq S$ can break off without decreasing the utility of at least one agent from $S \setminus S'$. Therefore, P is in the CSC. □

6 Conclusions and Future Work

In this paper, we initiate the study of ASHGs with a bounded coalition size. We provide MnM, an approximation algorithm for maximizing the utilitarian social welfare and study the computational aspects of the core, the SC, and the CSC. We note that MnM can be improved such that it finds a partition that is in the CSC while maintaining its approximation ratio. This is done by running MnM and iteratively joining together any two coalitions that improve the social welfare (without violating the size constraint).

There are several interesting directions for future work. Since we show that the MaxUtil problem cannot be computed in polynomial time (unless $P = NP$), it will be interesting to investigate some variants. For example, the problem of finding a k-bounded partition such that each agent will be matched with at least one friend in its coalition. Another interesting research direction is to incorporate skills in our model, motivated by coalitional skill games [3]. That is, each agent has a set of skills, and each coalition is required to have at least one agent that acquires each of the skills.

Acknowledgments

This research has been partly supported by the Ministry of Science, Technology & Space (MOST), Israel.

References

[1] Haris Aziz, Felix Brandt, and Hans Georg Seedig. Computing desirable partitions in additively separable hedonic games. *Artificial Intelligence*, 195:316–334, 2013.

[2] Haris Aziz, Rahul Savani, and Hervé Moulin. Hedonic games. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational Social Choice*, pages 356–376. Cambridge University Press, 2016.

[3] Yoram Bachrach and Jeffrey S Rosenschein. Coalitional skill games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1023–1030, 2008.

[4] Yoram Bachrach, Pushmeet Kohli, Vladimir Kolmogorov, and Morteza Zadimoghaddam. Optimal coalition structure generation in cooperative graph games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 27(1):81–87, Jun. 2013. doi: 10.1609/aaai.v27i1.8653. URL https://ojs.aaai.org/index.php/AAAI/article/view/8653.

[5] Coralio Ballester. Np-completeness in hedonic games. *Games and Economic Behavior*, 49(1): 1–30, 2004.

[6] Suryapratim Banerjee, Hideo Konishi, and Tayfun Sönmez. Core in a simple coalition formation game. *Social Choice and Welfare*, 18(1):135–153, 2001.

[7] Vittorio Bilò, Angelo Fanelli, Michele Flammini, Gianpiero Monaco, and Luca Moscardelli. Optimality and nash stability in additively separable generalized group activity selection problems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 102–108, 2019.

[8] Vittorio Bilò, Gianpiero Monaco, and Luca Moscardelli. Hedonic games with fixed-size coalitions. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*, volume 36, pages 9287–9295, 2022.

[9] Anna Bogomolnaia and Matthew O Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2):201–230, 2002.

[10] Katarı Cechlárová, Antonio Romero-Medina, et al. Stability in coalition formation games. *International Journal of Game Theory*, 29(4):487–494, 2001.

[11] Ágnes Cseh, Tamás Fleiner, and Petra Harján. Pareto optimal coalitions of fixed size. *arXiv preprint arXiv:1901.06737*, 2019.

[12] Xiaotie Deng and Christos H Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of operations research*, 19(2):257–266, 1994.

[13] Jacques H Dreze and Joseph Greenberg. Hedonic coalitions: Optimality and stability. *Econometrica: Journal of the Econometric Society*, pages 987–1003, 1980.

[14] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. doi: 10.4153/CJM-1965-045-4.

[15] Angelo Fanelli, Gianpiero Monaco, and Luca Moscardelli. Relaxed core stability in fractional hedonic games. In *Thirtieth International Joint Conference on Artificial Intelligence {IJCAI-21}*, pages 182–188. International Joint Conferences on Artificial Intelligence Organization ..., 2021.

[16] Michele Flammini, Gianpiero Monaco, Luca Moscardelli, Mordechai Shalom, and Shmuel Zaks. On the online coalition structure generation problem. *Journal of Artificial Intelligence Research*, 72:1215–1250, 2021.

[17] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[18] Lily Li, Evi Micha, Aleksandar Nikolov, and Nisarg Shah. Partitioning friends fairly. In *Proceedings of the 37th AAAI conference on Artificial Intelligence*, 2023.

[19] E. R. Love. 64.4 some logarithm inequalities. *The Mathematical Gazette*, 64(427):55–57, 1980. ISSN 00255572. URL http://www.jstor.org/stable/3615890.

[20] Michael Maschler, Bezalel Peleg, and Lloyd S Shapley. Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of operations research*, 4(4):303–338, 1979.

[21] Martin Olsen. Nash stability in additively separable hedonic games and community structures. *Theory of Computing Systems*, 45(4):917–925, 2009.

[22] Lloyd S Shapley and Martin Shubik. Quasi-cores in a monetary economy with nonconvex preferences. *Econometrica: Journal of the Econometric Society*, pages 805–827, 1966.

[23] Liat Sless, Noam Hazon, Sarit Kraus, and Michael Wooldridge. Forming k coalitions and facilitating relationships in social networks. *Artificial Intelligence*, 259:217–245, 2018.

[24] Shao-Chin Sung and Dinko Dimitrov. Computational complexity in additive hedonic games. *European Journal of Operational Research*, 203(3):635–639, 2010.

[25] Mason Wright and Yevgeniy Vorobeychik. Mechanism design for team formation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1):1050–1056, Feb. 2015. URL https://ojs.aaai.org/index.php/AAAI/article/view/9326.

Chaya Levinger
Ariel University
Ariel, Israel
Email: chayal@ariel.ac.il

Amos Azaria
Ariel University
Ariel, Israel
Email: amos.azaria@ariel.ac.il

Noam Hazon
Ariel University
Ariel, Israel
Email: noamh@ariel.ac.il

A Deferred Proofs

A.1 Proof of Lemma 1

Proof. Clearly, $Cliques_k$ is NP for every k. We use induction to show that any $Cliques_k$ is NP-Hard for every $k \geq 3$. $Cliques_3$ is known as the 'partition into triangles' problem, which was shown to be NP-Complete [17]. Given that $Cliques_k$ is NP-Hard we show that $Cliques_{k+1}$ is also NP-Hard. Given an instance of the $Cliques_k$ on a graph $G(V, E)$, we construct the following instance. We build a graph $G'(V', E')$, in which we add a set of nodes $\hat{V} = \hat{v}_1, ..., \hat{v}_{\frac{|V|}{k}}$, i.e., $V' = V \cup \hat{V}$. If $e \in E$ then $e \in E'$, and for every $v \in V, \hat{v} \in \hat{V}$ we add $(v, \hat{v})$ to E'. Clearly, V can be partitioned into disjoint cliques with exactly k vertices if and only if V' can be partitioned into disjoint cliques with exactly $k + 1$ vertices. □

A.2 Proof of Theorem 2

Proof. Clearly the problem is NP, since if we are given a partition P, a limit k, and the value υ, we can easily check that $\forall S \in P$, $|S| \leq k$ and that $u(P) \geq \upsilon$ in polynomial time. For the hardness proof, we use the $Cliques_k$ problem. Given an instance of $Cliques_k$ on an unweighted graph $G(V, E)$, we use the same graph with the same k and $\upsilon = |V|(k-1)$ as an instance to the MaxUtil problem. Clearly, V can be partitioned into disjoint cliques with exactly k vertices if and only if there exist a k-bounded partition P such that $u(P) = \upsilon$. □

A.3 Proof of Theorem 6

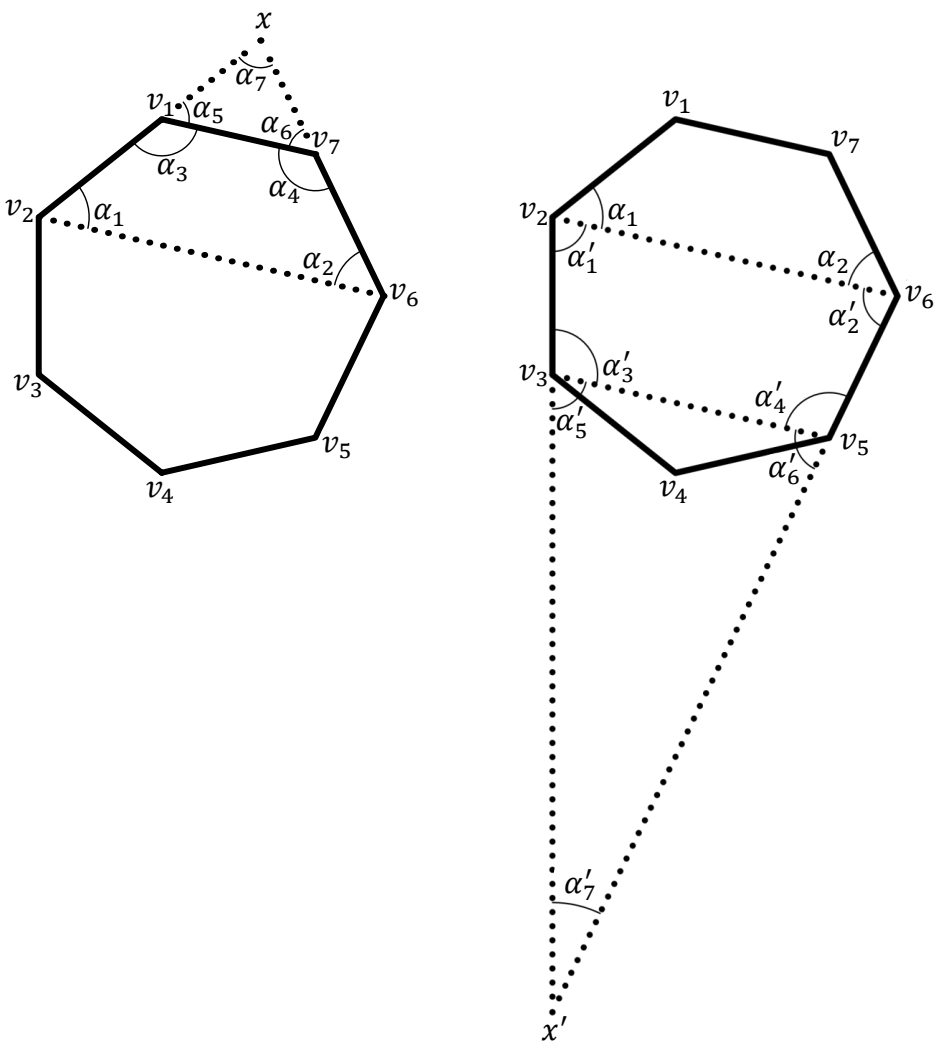
In order to prove Theorem 6, we must first prove a number of lemmas. Specifically, in Lemma 12 we characterize all the parallel lines in regular polygons with an odd number of sides, and in Lemma 13 we characterize all the parallel lines in regular polygons with an even number of sides.

Lemma 12. *Let $P = (v_1, ..., v_n)$ be a regular polygon, n is odd and $n > 3$, such that $v_1, ..., v_n$ are its vertices. For each side of P there are $\frac{n-1}{2}$ parallel lines that connect two vertices.*

Proof. Let $(v_1, v_2), (v_2, v_3), ..., (v_n, v_1)$ be the sides of the polygon P, and let $v_i v_j$ be the line that connects v_i to v_j. We denote by $d(p, p')$ the Euclidean distance between the points p and p'. We first show that $v_1 v_n$ is parallel to $v_2 v_{n-1}$. Since the sides (v_1, v_2) and (v_{n-1}, v_n) are not parallel, there exists a point, denoted by x, in which their corresponding lines intersect. Let $\alpha_1 = \sphericalangle v_1 v_2 v_{n-1}$, $\alpha_2 = \sphericalangle v_n v_{n-1} v_2$, $\alpha_3 = \sphericalangle v_n v_1 v_2$, $\alpha_4 = \sphericalangle v_1 v_n v_{n-1}$, $\alpha_5 = \sphericalangle x v_1 v_n$, $\alpha_6 = \sphericalangle x v_n v_1$, $\alpha_7 = \sphericalangle v_1 x v_n$ (Figure 4a demonstrates our notation of the angles for a polygon with 7 vertices). P is a regular polygon, and thus $\alpha_3 = \alpha_4$. Next, $\alpha_5 = \alpha_6 = 180^\circ - \alpha_3$. Thus, $d(v_1, x) = d(v_n, x)$ and $\alpha_5 = \frac{180^\circ - \alpha_7}{2}$. Now, since $d(v_2, v_1)$= $d(v_{n-1}, v_n)$, it holds that $d(v_2, x) = d(v_{n-1}, x)$. Therefore, $\alpha_1 = \alpha_2 = \frac{180^\circ - \alpha_7}{2} = \alpha_5$. Finally, since α_1 and α_5 are corresponding angles and $\alpha_1 = \alpha_5$, the lines $v_1 v_n$ and $v_2 v_{n-1}$ are parallel.

Note that since $\alpha_1 = \alpha_2$, also $\sphericalangle v_3 v_2 v_{n-1} = \sphericalangle v_{n-2} v_{n-1} v_2$. Now, if $n \geq 9$, a similar argument can show that $v_3 v_{n-2}$ is parallel to $v_2 v_{n-1}$. More generally, depending on n, this argument repeats t times, $t = \lfloor \frac{n}{4} \rfloor$. Namely, for any $i \in 1, 2, ..., t$ we show that the lines $v_i v_{n-i+1}, v_{i+1} v_{n-i}$ are parallel.

We now show that if $n \geq 7$, the lines $v_{t+1} v_{n-t}, v_{t+2} v_{n-t-1}$ are parallel. Since the sides (v_{t+1}, v_{t+2}) and (v_{n-t}, v_{n-t-1}) are not parallel, there exists a point, denoted by x', in which their corresponding lines intersect. Let $\alpha'_1 = \sphericalangle v_{t+2} v_{t+1} v_{n-t}$, $\alpha'_2 = \sphericalangle v_{t+1} v_{n-t} v_{n-t-1}$, $\alpha'_3 = \sphericalangle v_{t+1} v_{t+2} v_{n-t-1}$, $\alpha'_4 = \sphericalangle v_{n-t} v_{n-t-1} v_{t+2}$, $\alpha'_5 = \sphericalangle x' v_{t+2} v_{n-t-1}$, $\alpha'_6 = \sphericalangle x' v_{n-t-1} v_{t+2}$, $\alpha'_7 = \sphericalangle v_{t+2} x' v_{n-t-1}$ (Figure 4b demonstrates our notation of the angles for a polygon with 7 vertices. Note that for $n = 7$, t equals 1.). Since P is a regular polygon and $\alpha_1 = \alpha_2$, then $\alpha'_1 = \alpha'_2$. Next, since $\alpha'_1 = \alpha'_2$, then $d(v_{t+1}, x') = d(v_{n-t}, x')$ and $\alpha'_1 = \frac{180^\circ - \alpha'_7}{2}$. Now, since $d(v_{t+1}, v_{t+2})$=



(a) (b)

Figure 4: An example for a polygon with 7 vertices

$d(v_{n-t}, v_{n-t-1})$, it holds that $d(v_{t+2}, x') = d(v_{n-t-1}, x')$. Therefore, $\alpha'_5 = \alpha'_6 = \frac{180° - \alpha'_7}{2} = \alpha'_1$. Finally, since α'_1 and α'_5 are corresponding angles and $\alpha'_1 = \alpha'_5$, the lines $v_{t+1}v_{n-t}$ and $v_{t+2}v_{n-t-1}$ are parallel.

Now, if $n \geq 11$, a similar argument can show that $v_{t+2}v_{n-t-1}$ is parallel to $v_{t+3}v_{n-t-2}$. More generally, depending on n, this argument repeats t' times, $t' = \lfloor \frac{n+1}{4} \rfloor - 1$. Namely, for any $i' \in 1, 2, ..., t'$ we show that the lines $v_{t+i'}v_{n-t-i'+1}, v_{t+i'+1}v_{n-t-i'}$ are parallel.

Finally, using the same arguments for all sides, we obtain that for each side of P there are $\frac{n-1}{2}$ parallel lines that connect two vertices.

□

Lemma 13. *Let $P = (v_1, ..., v_n)$ be a regular polygon, n is even and $n > 3$, such that $v_1, ..., v_n$ are its vertices. For each side of P there are $\frac{n}{2} - 1$ parallel lines that connect two vertices. In addition, for each line $v_i v_{((i+2) \mod n)}$ there are $\frac{n}{2} - 2$ parallel lines that connect two vertices.*

Proof. Obviously, every side of the polygon is parallel to another side. Now, using a similar argument to the one used in Lemma 12 we can show that (v_1, v_n) is parallel to $v_2 v_{n-1}$. This argument can be repeated $\lfloor \frac{n-2}{4} \rfloor$ times. In addition, the same argument shows that $v_{\frac{n}{2}} v_{\frac{n}{2}+1}$ is parallel to $v_{\frac{n}{2}-1} v_{\frac{n}{2}+2}$, and can be repeated $\lfloor \frac{n}{4} \rfloor - 1$ times. Since (v_1, v_n) is parallel to $v_{\frac{n}{2}} v_{\frac{n}{2}+1}$, all said lines are parallel. Clearly, the same arguments can be used for all sides. Overall, for each side there are $\frac{n}{2} - 1$ parallel lines that connect two vertices. Similar arguments can be used to show that for each line $v_i v_{((i+2) \mod n)}$ there are $\frac{n}{2} - 2$ parallel lines that connect two vertices. □

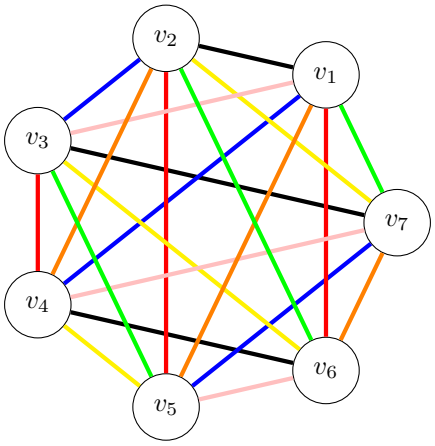


Figure 5: Division of all the edges in a graph with 7 vertices into 7 disjoint groups, using parallel lines.

We can deduce from Lemmas 12 and 13, that given a complete graph, the set of edges can be partitioned into n subsets, such that each subset is a matching.

Lemma 14. *Let $G(V, E)$ be a complete graph with n vertices, $n \geq 3$. E can be partition into n matchings.*

Proof. Note that G can be represented as a regular polygon $P = (v_1, ..., v_n)$. We first consider the case in which n is odd. The set $\{(v_1, v_n), (v_2, v_{n-1}), ..., (v_{\frac{n-1}{2}}, v_{\frac{n-1}{2}+2})\}$ is a valid matching, since all the edges are parallel to each other by Lemma 12. Therefore, for each side of P, we obtain a matching of size $\frac{n-1}{2}$. Since n is odd, there are no two parallel sides in P. Therefore, we obtain n disjoint matchings of size $\frac{n-1}{2}$ each, which totals at $\frac{n(n-1)}{2} = |E|$. An example for this partition for $n = 7$ is shown in Figure 5.

Next, we consider the case in which n is even. The set $\{(v_1, v_n), (v_2, v_{n-1}), ..., (v_{\frac{n}{2}}, v_{\frac{n}{2}+1})\}$ is a valid matching, since all the edges are parallel to each other by Lemma 13. Therefore, for each side of P we obtain a matching of size $\frac{n}{2}$. Now, since n is even, for every side of P there is exactly one other parallel side. Thus, we obtain $\frac{n}{2}$ disjoint matchings of size $\frac{n}{2}$. In addition, the set $\{(v_{n-1}, v_1), (v_{n-2}, v_2), ..., (v_{\frac{n}{2}+1}, v_{\frac{n}{2}-1})\}$ is a valid matching since all the edges are parallel to each other (by Lemma 13). Note that this matching does not include an edge in which v_n or $v_{\frac{n}{2}}$ is an endpoint. Thus, we obtain $\frac{n}{2}$ disjoint matchings of size $\frac{n-2}{2}$. Overall, we obtain $\frac{n}{2} \cdot \frac{n}{2} + \frac{n}{2} \cdot \frac{n-2}{2} = \frac{n}{2} \cdot (\frac{n}{2} + \frac{n-2}{2}) = \frac{n(n-1)}{2} = |E|$. An example for this partition for $n = 8$ is shown in Figure 6. □

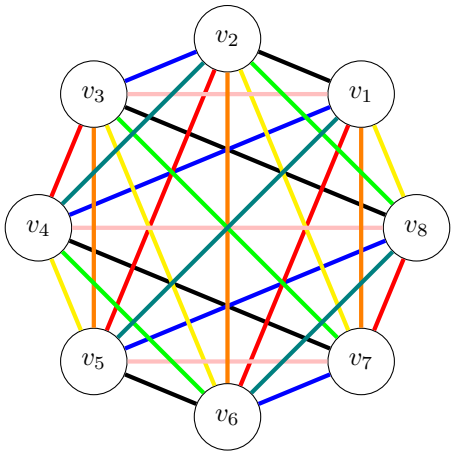


Figure 6: A partition of all the edges in a graph with 8 vertices into 8 disjoint groups, using parallel lines.

We are now ready to prove Theorem 6.

Proof. let $Opt = \{S_1, S_2, ...\}$ be an optimal k-bounded partition, and let M_i be a maximum weight matching for coalition S_i. Lemma 14 shows that each coalition S_i can be partitioned into $|S_i|$ disjoint matchings. In addition, for each i, $|S_i| \leq k$. Therefore, for each i, $\sum_{e \in M_i} w(e) \geq \frac{\sum_{v \in S_i} W(v, S_i)}{k}$. Let M be the maximum weight matching for G found by Algorithm 1 in its first step. Clearly, $\sum_{e \in M} w(e) \geq \sum_i \sum_{e \in M_i} w(e)$. In addition, for the k-bounded partition P that MnM returns, $u(P) \geq \sum_{e \in M} w(e)$. Therefore, Algorithm 1 provides a solution for the MaxUtil problem with an approximation ratio of $\frac{1}{k}$ for every $k \geq 3$, in the weighted setting. □

A.4 Proof of Theorem 9

Proof. Clearly, if the algorithm terminates, the k-bounded partition P is in the ϵ_m-core. We show that $u(P)$ always increases. The algorithm initiates a new iteration whenever $\forall v \in S, W(v, S) > \frac{u(v,P)}{\epsilon_m}$, which can happen only if S breaks off. When S breaks off, the social welfare decreases by at most $2 \cdot \sum_{v \in S} u(v, P)$, and increases by at least $\sum_{v \in S} (2 \cdot u(v, P) + 1)$. Since $\sum_{v \in S} (2 \cdot u(v, P) + 1) > 2 \cdot \sum_{v \in S} u(v, P)$, then $u(P)$ always increases. □

A.5 Proof of Theorem 10

Proof. We use a reduction from $Cliques_k$ for the hardness proof. Given an instance of the $Cliques_k$ on a graph $G(V, E)$, we construct the following instance. We build a graph $G'(V', E')$ such that V'

contains all the nodes from V. In addition, for every $v_x \in V$ we add the nodes $\hat{v}_x$ and $v_x^1, \ldots, v_x^{k-1}$ to V'. Now, E' contains all the edges of E, and for every $v_x \in V$ and $1 \leq i \leq k-1$ we add $(v_x, v_x^i), (v_x^i, \hat{v}_x)$ to E'. Finally, for every $v_x \in V$ and $1 \leq i, j \leq k-1$, $i \neq j$ we add (v_x^i, v_x^j) to E'. Figure 7 shows this construction for a specific v_x when $k = 4$. We first show that if G cannot be partitioned into disjoint cliques of size k, then the strict core is empty. Indeed, assume that G cannot be partitioned into disjoint cliques of size k, and let P be a k-bounded partition of V'. Then, there is at least one vertex $v_x \in V$ that belongs to a coalition $S \in P$, such that either:

1. $W(v_x, S) < k-1$, or
2. $v_x^i \in S$ for some i between 1 and $k-1$.

In case 1, the coalition $\{v_x, v_x^1, \ldots, v_x^{k-1}\}$ is a weakly blocking k-bounded coalition. In case 2, the coalition $\{\hat{v}_x, v_x^1, \ldots, v_x^{k-1}\}$ is a weakly blocking k-bounded coalition. Therefore, if the strict core is not empty, then G can be partitioned into disjoint cliques of size k.

We now show that if G can be partitioned into disjoint cliques of size k, then the strict core is not empty. Clearly, in this case G' can be partitioned into disjoint cliques of size k, and this partition is in the strict core. Therefore, if the strict core is empty, then G cannot be partitioned into disjoint cliques of size k. □

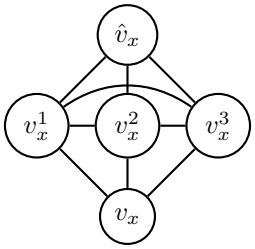


Figure 7: The reduction construction for a specific v_x, when $k = 4$.

B Additional Results

B.1 Tightness of the Approximation Ratio of the MnM Algorithm in the Unweighted Setting

We show that our approximation ratio in the unweighted setting is tight.

Theorem 15. *The approximation ratio of MnM for the MaxUtil problem in the unweighted setting is tight.*

Proof. Given $k > 2$, consider a complete graph of size $2k$. In this case, MnM finds a perfect matching in M_1, and thus the partition P returned by MnM contains k groups of 2 nodes. That is, $u(P) = 2k$. On the other hand, an optimal k-bounded partition Opt consists of 2 Cliques of size k, and thus $u(Opt) = 2k(k-1)$. That is, MnM provides an approximation of exactly $\frac{1}{k-1}$. □

Figure 8 presents a case where $k = 5$, and G is a complete graph with 10 nodes. Here, $P = \{\{v_1, v_2\}, \{v_3, v_4\}, \{v_5, v_6\}, \{v_7, v_8\}, \{v_9, v_{10}\}\}$, as shown in the dotted lines, and thus $u(P) = 10$. However, $Opt = \{\{v_1, v_2, v_3, v_4, v_5\}, \{v_6, v_7, v_8, v_9, v_{10}\}\}$, as shown in the blue lines, and $u(Opt) = 40$.

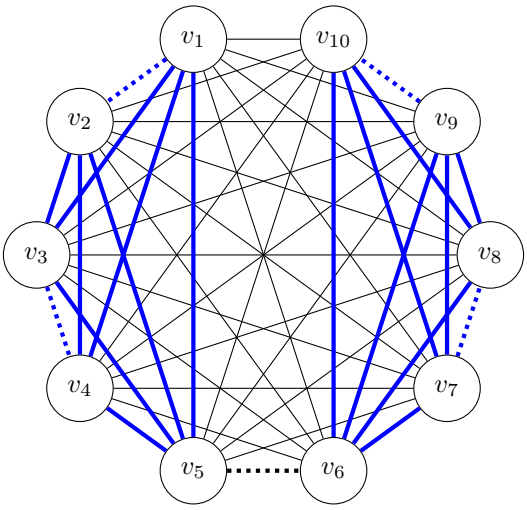


Figure 8: An example of a graph in which $k = 5$, and MnM achieves an approximation ratio of exactly $\frac{1}{4}$.

B.2 MaxUtil with Distributed Partition in Unweighted Setting

We analyze a procedure that attempts to model the behavior of the agents when there is no central mechanism that determines the partition. Assume that the agents are split up arbitrarily but maximally, i.e., in a way that no two coalitions can joint together such that the social welfare will be higher. We call this procedure $Arbmax$. Without loss of generality, we assume that every set $S \in Arbmax$ is a connected component. We show that $\frac{1}{k}$ is an upper bound on the approximation ratio that $Arbmax$ may guarantee.

Theorem 16. *For any k, $Arbmax$ cannot guarantee an approximation ratio better than $\frac{1}{k}$*

Proof. Given k, consider the following graph G. There are k distinguished nodes, $v_1, \ldots, v_k$, with the edges $(v_i, v_{i+1}) \in E$ for $i = 1, \ldots, k-1$. Each distinguished node v_i has $k-1$ additional neighbors that are connected only to v_i, i.e., v_i is the internal node of a star graph with $k-1$ leaves. Clearly, Opt consists of k coalitions, where each coalition consists of a star graph. Thus, $u(Opt) = 2k(k-1)$. On the other hand, $Arbmax$ may partition the graph such that the distinguished nodes $v_1, \ldots, v_k$ are in the same coalition. Since there are no edges between two undistinguished nodes, the social welfare of this partition is $2(k-1)$. Therefore, $Arbmax$ cannot guarantee an approximation ratio better than $\frac{1}{k}$. □

Figure 9 presents a case where $k = 5$, and $Arbmax$ may provide only a $\frac{1}{5}$ of the social welfare provided by an optimal solution. Here, $Arbmax$ may return the partition $P' = \{\{v_1, v_2, v_3, v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}, \{v_9\}, \{v_{10}\}, \{v_{11}\}, \{v_{12}\}, \{v_{13}\}, \{v_{14}\}, \{v_{15}\}, \{v_{16}\}, \{v_{17}\}, \{v_{18}\}, \{v_{19}\}, \{v_{20}\}, \{v_{21}\}, \{v_{22}\}, \{v_{23}\}, \{v_{24}\}, \{v_{25}\}\}$ and thus $u(P') = 8$, while $Opt = \{\{v_1, v_6, v_7, v_8, v_9\}, \{v_2, v_{10}, v_{11}, v_{12}, v_{13}\}, \{v_3, v_{14}, v_{15}, v_{16}, v_{17}\}, \{v_4, v_{18}, v_{19}, v_{20}, v_{21}\}, \{v_5, v_{22}, v_{23}, v_{24}, v_{25}\}\}$ and therefore $u(Opt) = 40$.

B.3 Worst Case Example for the MnM Algorithm in the Weighted Setting

We show that in the weighted setting, MnM cannot guarantee a better approximation ratio. Indeed, Figure 10 shows an example where $k = 3$ and MnM $= \{\{v_1, v_2\}, \{v_3, v_4\}, \{v_5, v_6\}\}$. In this case, the outcome of MnM provides only a $\frac{2+\epsilon}{6}$ of the social welfare provided by the optimal solution $\{\{v_1, v_2, v_3\}, \{v_4, v_5, v_6\}\}$. Therefore, for any $\epsilon > 0$, when $k = 3$, MnM cannot provide an approximation that is better than $\frac{1}{k} + \epsilon$.

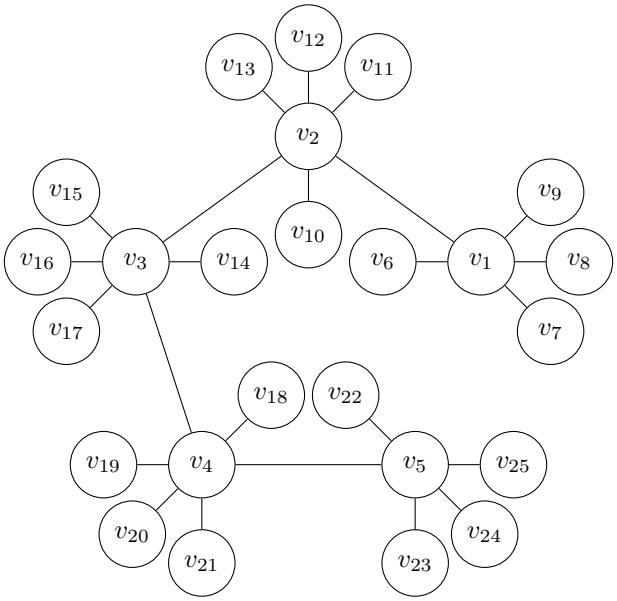


Figure 9: A case where $k = 5$, and $Arbmax$ may provide only a $\frac{1}{5}$ of the social welfare provided by an optimal solution.

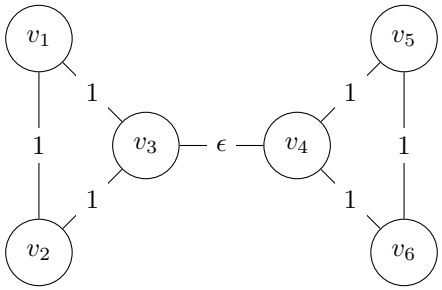


Figure 10: A case where $k = 3$, and MnM provides only a $\frac{2+\epsilon}{6}$ of the social welfare provided by an optimal solution.

B.4 Nash Stablity

Due to [9], for any (G, k) a Nash stable k-bounded partition exists. In this section, we present Algorithm 5, a polynomial time algorithm that finds such a partition in the unweighted setting. The algorithm begins with all agents in singletons and iteratively considers for each agent whether it may benefit from leaving her coalition and joining a coalition of size of at most $k-1$.

Algorithm 5: Finding a Nash stable k-bounded partition

```
1 Input: A graph G(V, E) and a limit k
  Result: A k-bounded Nash Stable partition P of V.
2 P ← {{v} for every v ∈ V}
3 outerLoop:
4 for v ∈ V do
5 |   for S ∈ P do
6 |   |   if W(v, S ∪ {v}) > u(v, P) AND |S| ≤ k − 1 then
7 |   |   |   P ← P^{−S∪{v}}
8 |   |   |   goto outerLoop
9 return P
```

Theorem 17. *There always exists a k-bounded Nash stable partition, and it can be found in polynomial time.*

Proof. Consider Algorithm 5. Clearly, when Algorithm 5 terminates, the partition P is a k-bounded Nash Stable partition. We now show that Algorithm 5 must always terminate, and it runs in polynomial time. Returning to line 3 occurs only when the if statement in line 6 is true, which entails that $u(P)$ has increased. Since any increase in $u(P)$ must be by multiples of 2, and since $u(P)$ is bounded by $2|E|$, the algorithm must terminate after at most $|E|$ iterations. □