Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Optimal Coalition Structure Generation in Cooperative Graph Games

Yoram Bachrach[†], **Pushmeet Kohli**[†], **Vladimir Kolmogorov**[‡] and **Morteza Zadimoghaddam**[§]
† Microsoft Research, Cambridge, UK
‡ Institute of Science and Technology (IST), Austria
§ MIT, Cambridge, MA, USA

Abstract

Representation languages for coalitional games are a key research area in algorithmic game theory. There is an inherent tradeoff between how general a language is, allowing it to capture more elaborate games, and how hard it is computationally to optimize and solve such games. One prominent such language is the simple yet expressive *Weighted Graph Games (WGGs)* representation (Deng and Papadimitriou 1994), which maintains knowledge about synergies between agents in the form of an edge weighted graph.

We consider the problem of finding the optimal coalition structure in WGGs. The agents in such games are vertices in a graph, and the value of a coalition is the sum of the weights of the edges present between coalition members. The *optimal coalition structure* is a partition of the agents to coalitions, that maximizes the sum of utilities obtained by the coalitions. We show that finding the optimal coalition structure is not only hard for general graphs, but is also intractable for restricted families such as planar graphs which are amenable for many other combinatorial problems. We then provide algorithms with constant factor approximations for planar, minor-free and bounded degree graphs.

Introduction

Consider a set of agents who can work in teams. Some agents work well together, while others find it hard to do so. When two agents work well together, a team which contains both of them can achieve better results due to the synergy between them. However, when agents find it hard to work together, a team that contains both agents has a reduced utility due to their inability to cooperate, and may perform better when one of them is removed. How should we best partition agents into teams to maximize the total utility generated?

Cooperation is a central issue in algorithmic game theory, and *cooperative games* are very useful for modeling team formation and negotiation in many domains. In such games, agents form coalitions to pursue a joint cause, and must decide how to split the gains they derive as a group. Much of previous literature explores settings where only one coalition can be formed. However, in many scenarios agents can form *multiple* disjoint coalitions, where each coalition can obtain its profits independently. An important goal is to partition the agents in a way that maximizes the total value gained by all coalitions, i.e. the social welfare. This problem is known as *finding the optimal coalitional structure*.

Cooperative game theory provides tools to reason about how to best partition the agents into team or how the utility generated by an agent team should be allocated to its members. The key piece of information used by such game theortic tools is the amount of utility any team of agents could potentially generate. Since an agent team, sometimes called a *coalition* is simply a subset of agents, the number of different coalitions is exponential in the number of agents. The mapping between any agent coalition and the utility it can generate lies at the heart of any cooperative game, and is called the *characteristic function* of the game. One way to represent the characteristic function is by simply listing down this utility for any possible coalition. Generally any coalition may have a different value, so such a naïve representation requires storage exponential in the number of agents, emphasizing the need for a *succinct representation*.

In many domains it is possible to use *knowledge* about *specific features* of the domains and provide a more succint representation of the characteristic function. However, even for such succint representations, reasoning about the game may still be computationally hard. Previous work in algorithmic game theory has examined many such representation languages for cooperative games and the computational complexity of calculating various solutions in them. For general surveys of such representations, see (Chalkiadakis, Elkind, and Wooldridge 2011; Shoham and Leyton-Brown 2009; Bilbao 2000). Some approaches guarantee a polynomial description, but only represent restricted games (Peleg and Sudholter 2007; Deng and Papadimitriou 1994). Others can represent any game, but require exponential storage in the worst case (Ieong and Shoham 2006; Bachrach et al. 2010). We examine coalition structures in the prominent *Weighted Graph Games (WGG)* model of (Deng and Papadimitriou 1994). In WGGs, agent synergies are expressed using a graph, where agents are vertices and the weight of an edge connecting two agents expresses how well they work together. A positive weight indicates they can coordinate well, yielding a positive contribution to a coalition containing both. The edge's weight expresses how much utility can be derived from the cooperation of the two agents. A negative weight indicates the agents do not work well to-

gether, diminishing the utility of a team containing both; the edge's weight expresses the reduction in utility of a coalition that contains both agents. This graph representation allows expressing synergies in coalitions that agents form. [1]

Our contribution: We study optimal coalition structures in WGGs. We prove that finding the optimal coalition structure in WGGs is hard even for restricted families such as planar graphs. We provide constant factor approximation algorithms for planar, minor-free and bounded degree graphs.

Note that our objective function coincides with the *Correlation Clustering* functional (Bansal, Blum, and Chawla 2004) **up to an additive constant**. Due to this additive shift existing approximability results for Correlation Clustering (Tan 2008; Mitra and Samal 2009) do not translate to our problem. We believe that our additive normalization is quite natural in our context. To our knowledge, we are the first to study approximation schemes for this functional.

Preliminaries: A transferable utility (TU) coalitional game is composed of a set of n agents, I, and a characteristic function mapping any subset (coalition) of the agents to a rational value $v : 2^I \rightarrow \mathbb{Q}$, indicating the total utility these agents achieve together. We follow the *Coalition Structure (CS)* model of (Aumann and Dreze 1974), where agents can form several teams *simultaneously*. A coalition structure is a partition of the agents into disjoint coalitions. Formally, $CS = (C^1, \ldots, C^l)$ is a *coalition structure* over I if $\cup_{i=1}^{l} C^i = I$ and $C^i \cap C^j = \emptyset$ for all $i \neq j$; The set $\mathcal{CS}(I)$ denotes all possible coalition structures on I.

We overload notation and denote $v(CS) = \sum_{C^j \in CS} v(C^j)$. We focus on the coalitional structure generation problem, of finding the *optimal coalition structure* CS^*, with maximal value. Given a game $\langle I, v \rangle$, an *optimal* coalition structure $CS^* \in \mathcal{CS}(I)$ is a partition that maximizes welfare, i.e. for any other $CS \in \mathcal{CS}(I)$ we have $v(CS) \leq v(CS^*)$. We denote the problem of finding an optimal coalition structure as *OPT-CS*. OPT-CS is equivalent to a complete set partitioning problem where all disjoint subsets of the set of all agents are possible, which was was studied in (Yun Yeh 1986) and shown to be NP-hard. However, we focus on OPT-CS where inputs are restricted to WGGs (Deng and Papadimitriou 1994).

We now review the *Weighted Graph Games (WGGs)* model (Deng and Papadimitriou 1994).

Definition 1. *WGGs are games played over a graph $G = \langle V, E \rangle$, with edge weights $w : E \rightarrow \mathbb{Q}$. The agents are the vertices, so $I = V$, and the characteristic function is the sum of the weights on the graph induced by the coalition. Given a coalition $C \subseteq V$, we denote the edges induced by the coalition as $E_C = \{e = (u, v) \in E | u, v \in C\}$. The characteristic function is $v(C) = \sum_{e \in E_C} w(e)$.*

As noted in (Deng and Papadimitriou 1994) WGG is an incomplete representation language, and some characteristic functions cannot be represented in it. We allow at most one edge between any two vertices (parallel edges may be merged, summing the weights).

[1] Group buying sites (e.g. LivingSocial and Groupon) reward social recommendations, so WGGs can capture synergies from including enough friends of a consumer to make her buy a good.

Finding the Optimal Coalitional Structure

We first formally define the OPT-CS problem.

Definition 2 (OPT-CS-WGG). *Given a WGG $\langle I, v \rangle$ and a rational number r, test if the value of the optimal coalitional structure for this game is at least r, i.e. if there is $CS \in \mathcal{CS}(I)$ such that $v(CS) \geq r$. We denote an optimal structure as $CS^* \in \arg\max_{CS \in \mathcal{CS}(I)} v(CS)$.*

Hardness Results

We first discuss why OPT-CS-WGG is hard. Independently of us (Voice, Polukarov, and Jennings 2012) examine a related coalition structure generation problem for a graph representation. Their representation is combinatorially richer, so their hardness results do not generally carry over to our simpler WGG representation. It is easy to show hardness for OPT-CS-WGG over general graphs using a reduction from Independent Set (IS). IS is the problem of testing if there is a subset of vertices of size k with no edges between them in an input graph $G = \langle V, E \rangle$. Theorem 3 is a stronger result, so we only provide a brief sketch of the proof.

Theorem 1. *OPT-CS-WGG is $\mathcal{NP}$-complete, and assuming* P $\neq$ NP*, there is no polynomial time $O(n^{1/2-\epsilon})$-approximation algorithm for it where n is the number of vertices and ϵ is any positive constant. There is no polynomial time $O(n^{1-\epsilon})$-approximation algorithm for this problem unless* NP = ZPP.

Proof. We reduce an IS instance $\langle G = \langle V, E \rangle, k \rangle$ to an OPT-CS-WGG instance. The reduced graph $G' = \langle V', E' \rangle$ has all vertices and edges of G (so $V \subseteq V'$ and $E \subseteq E'$), and an additional vertex s. The weights of the original edges are all set to be $-k$ (where $k > |V|$). Vertex s is connected to each vertex in $v \in V$ with an edge e of weight $w(e) = 1$. We show G has an independent set of size k iff there exists a partition for G' with value k. If G has an independent set S with size k, we can create a coalition structure CS with value k: we put s and the k vertices of S in one coalition. For any other $v \notin S$, we make a separate coalition containing only v (with zero value). The value of the entire coalition structure is the value of the coalition with s. There are no negative edges in this coalition, as S is an independent set in G. There are k edges with weight $+1$ in this coalition, so the value of the coalition structure is k. If the optimal coalition structure has value k' in graph G', we can find an independent set of size k' in G. The absolute value of a negative edge is more than the sum of all positive edge weights, as there are $n = |V|$ positive edges with weight $+1$. We can put each vertex in a separate coalition and achieve a value of zero. Thus in the optimal coalition structure we never put a negative edge in a coalition, as this obtains a negative value. Hence, the value of the optimal coalition structure is the number of positive edges we get through the coalitions. All positive edges are connected to s, so the value of the coalitions is the number of vertices we put in s's coalition. We can safely put the remaining vertices in separate coalitions, as there are no positive edges between them. In the optimal solution the vertices we put in s's coalition have no negative edges between them, forming an independent set.

This is a parsimonious reduction between IS and OPT-CS-WGG. Håstad proved that there is no polynomial time $O(n^{1/2-\epsilon})$-approximation algorithm for IS assuming P $\neq$ NP, and no polynomial time $O(n^{1-\epsilon})$-approximation algorithm for IS unless NP = ZPP (Håstad 1996). □

The above result shows that OPT-CS-WGG has no good approximation for general graphs. The hardness might seem to come from having to avoid all negative edges, as we can make the weights of the negative edges very low to make sure that they are not present in the optimal solution. However, we show OPT-CS-WGG is hard and inapproximable even when all weights have the same absolute value, using a reduction from IS to the problem of OPT-CS-WGG where all edges are either $+1$ or -1, denoted OPT-CS-WGG± 1.

Theorem 2. *OPT-CS-WGG± 1 is $\mathcal{NP}$-complete. Assuming* P $\neq$ NP*, there is no polynomial time $O(n^{1/2-\epsilon})$-approximation algorithm for it where n is the number of vertices of the graph, and ϵ is any positive constant. There is no polynomial time $O(n^{1-\epsilon})$-approximation algorithm for this problem unless* NP = ZPP.

Proof. Similarly to the proof of Theorem 1, we reduce an Independent Set instance $G = \langle V, E\rangle$ to a OPT-CS-WGG instance. The reduced graph $G' = \langle V', E'\rangle$ contains all the vertices and edges of G (so $V \subseteq V'$ and $E \subseteq E'$), and one additional vertex s. The weights of the original edges are identical, and are all $w(e) = -1$ (this is where the reduction differs from that of Theorem 1). Vertex s is connected to any vertex in $v \in V$ with an edge e with weight $w(e) = +1$.

If the optimal coalition structure has value k' in graph G', we show we can find an independent set of size at least $k'/9$ in G. As in the previous reduction, all positive edges are incident to vertex s. Thus every vertex is either in the coalition of vertex s or in a separate single-vertex coalition. Let S be the set of vertices from graph G in the coalition of vertex s. The value of coalition S is thus the value of the entire coalition structure. Suppose there are a vertices in S, and b negative edges between vertices of S. In this case, the sum of all weights of positive edges for S is equal to a, and the sum of all weights of negative edges in S is $-b$. Therefore, $k' = a - b$, so $a > k'$. Also note that $b < a$.

Now consider the subgraph $G[S]$. The number of edges in this graph, b, is not more than the number of vertices a. Therefore, the average degree is at most 2 in this subgraph. Thus, the number of vertices with degree at least 3 in this subgraph is not more than $2a/3$, so there are at least $a/3$ vertices with degree at most 2.

Let S' be the set of vertices with degree at most 2 in the subgraph $G[S]$. Now consider the subgraph $G[S']$. This subgraph has at least $a/3$ vertices, each with a degree at most 2. We can pick $1/3$ of the vertices of this subgraph as an independent set: we just pick a vertex arbitrarily, put it in our independent set and remove its neighbors, which are no more than two vertices. Thus, in the worst case, for every three vertices, we choose one for our independent set. This way, we can find an independent set of size at least $a/9 \geq k'/9$, so our reduction loses a factor of at most 9, but the same hardness results hold asymptotically. □

Though (Voice, Polukarov, and Jennings 2012) examine a richer domain, their proof that generating the optimal coalition structure for planar does carry over to our restricted WGG setting (an alternative proof uses an intricate reduction from Independent Set). This yields Theorem 3.

Theorem 3. *OPT-CS-WGG in planar graphs is $\mathcal{NP}$-complete.*

Proof. See (Voice, Polukarov, and Jennings 2012). □

Exact Algorithms for Bounded Treewidth Graphs

For completeness, we first consider OPT-CS-WGG restricted to graphs of bounded treewidth. Lemma 1 is a special case of Lemma 2 from (Cowans and Szummer 2005).[2]

Lemma 1. *OPT-CS-WGG with inputs restricted to trees (or forests) is in $\mathcal{P}$.*

Proof. Let $P \subseteq E$ be the set of edges with positive weights, and $N \subseteq E$ be the edges with negative weights. For an edge subset A we denote its total weight as $w(A) = \sum_{e\in A} w(e)$. Note that for any structure CS we have $v(CS) \leq w(P)$. We show that for trees the optimal structure CS^* has $v(CS^*) = w(P)$. A very simple partitioning to two sub coalitions X, Y suffices for this. We begin with an arbitrary leaf v, and put it in one of the sub coalitions, so $v \in A$. For any neighbor u of v, so $e = (v, u) \in E$, we choose a sub coalition for u based on $w(e)$. If $w(e) > 0$ we put u in the same sub coalition as v, and if $w(e) < 0$ we put u in the other sub coalition. We then continue the process from the vertex u, until we deplete all the vertices in the graph. Since the graph is a tree, any edge with negative weight has one vertex in X and the other in Y, and is not counted for the value of the coalition structure, while every positive weight edge has both its vertices in the same sub coalition, and is counted for the value of the coalition structure. Thus we have $v(CS^*) = w(P)$. The partitioning can be done using a simple breadth first search, in polynomial time of $O(|V| + |E|)$. □

Lemma 2 ((Cowans and Szummer 2005)). *For k-bounded treewidth graphs (constant k), OPT-CS-WGG is in $\mathcal{P}$.*

Constant Factor Approximation for Planar Graphs

We provide a polynomial time constant approximation algorithm for planar graph games. Planar graphs model many real-world domains. For example, consider coalitions between countries. In many domains, synergies would only exist between neighbouring countries. We can define a graph game where countries have an edge between them only if they are neighbours, resulting in a *planar* graph. Our method achieves a coalition structure avoiding all negative edges and gains a constant portion of the weights of the positive edges. The optimal value is at most the sum of the positive weights,

[2] We decided to include Lemma 1 since (i) its proof is much simpler than that of Lemma 2; (ii) it provides a distributed (local) algorithm for partitioning. Two neighbors are in the same coalition if and only if their connecting edge has positive weight; (iii) it shows that for forests, there is a simple solution that achieves all positive edges, and avoids all negative edges.

yielding a constant approximation. First we define feasible sets which play an important role in our algorithm.

Definition 3. *Given a planar graph G, denote by E^+ the set of positive edges, and by E^- the set of negative edges. A subset $E' \subseteq E^+$ is* a feasible set *iff there is a partition P of vertices such that any edge $e \in E'$ is contained in one part of the partition, while all negative edges are cross-part edges. We say P achieves E'.*

We find partitions that achieve feasible sets $E_1, E_2, \cdots, E_k$ (each a subset of E^+), whose union $\cup_{i=1}^{k} E_i$ is E^+. Each positive edge is thus achieved by at least one of these k partitions. The value of partition i is at least $\sum_{e \in E_i} w(e)$ as we avoid all negative edges in our partitions, making the value of a partition be the sum of the positive weights achieved by it. The union of these k feasible sets is the set of all positive edges. Thus the sum of the values of the k partitions is at least the sum of all positive weights. Picking the maximal value partition, we obtain a k-approximation algorithm. Our algorithm finds *few* such partitions that still *cover all positive edges*. In our algorithm we present a constant number of these partitions (feasible sets). We first discuss basic feasible sets that we use in it. The first building block is a matching.

Lemma 3. *Every matching $M \subseteq E^+$ is a feasible set. In other words, there is a partition that avoids all negative edges, and achieves edges of M.*

Proof. Let $e_1, e_2, \cdots, e_a$ be the edges of a matching. We build a partition of the vertices. We have a clusters for the a edges of our matching. We put both endpoints of e_i in cluster i. There are $n - 2a$ remaining vertices as well. We put them in $n - 2a$ separate single-vertex clusters, so we achieve the edges of M in this partition. We show we avoid all negative edges. Suppose not, so there is a negative edge $e'(u, v)$ such that u and v are in the same cluster. Thus u and v are endpoints of an edge in the matching as well, so there are two edges between u and v in to contradiction to our assumption of having no parallel edges. □

The second block is more elaborate. We show that the union of vertex-disjoint stars can be covered using at most three feasible sets. A star is a subgraph of several edges that all share one endpoint called center vertex. In other words, a star is a vertex with some edges to some other vertices, and there are no other edges in the star between vertices.

Lemma 4. *We can cover a union of several vertex-disjoint stars using at most 3 feasible sets.*

Proof. Assume there are l stars with center vertices $v_1, v_2, \cdots, v_l$. In the star i, vertex v_i has edges to vertex set S_i. The sets $S_1, S_2, \cdots, S_l$ are disjoint and do not contain any of the center vertices. Since G is planar we can find a proper four-coloring for it. Consider vertex v_i. Non of the vertices in set S_i has the color of vertex v_i, so the vertices of set S_i are colored using only three colors. Without loss of generality, assume they are colored with colors $1, 2, 3$. Let $S_{i,1}$ be the set of vertices in S_i with color one. Similarly we define sets $S_{i,2}$ and $S_{i,3}$. We do this for all center vertices. Note that there is no edge inside set $S_{i,j}$ for $1 \leq i \leq l$, and $1 \leq j \leq 3$, but there may be edges between different sets. We define the first feasible set. We put v_i and all vertices of $S_{i,1}$ in one cluster, and we do this for all center vertices, so for each center vertex, we have a separate cluster. All remaining vertices of the graph go to separate single-vertex clusters. We achieve the edges between vertex v_i, and all vertices in $S_{i,1}$ for $1 \leq i \leq l$. We avoid all negative edges as there is no edge inside set $S_{i,1}$, and there is no negative edge between vertex v_i, and set $S_{i,1}$. Similarly we use two other partitions to get the edges between v_i and sets $S_{i,2}$ and $S_{i,3}$. Thus we cover all these edges using three feasible sets. □

Lemma 5. *The edges of a forest can be covered using 6 feasible sets.*

Proof. We show the proof for one tree. The same should be done for other trees. Make the tree T rooted at some arbitrary vertex r. Now every vertex in the tree has a unique path to r. We color the edges of this tree using two colors, red and blue. The edges that have an odd distance to r are colored red, and the edges with even distance to r are blue. So the first level of edges that are adjacent to r are colored blue, the next level edges are red, and so on. So we start from r, and alternatively color edges blue and red. Considering the subgraph of blue edges, we cannot find a path of length four (3 edges) in it, and we know that there is no cycle in the graph. Thus, these blue edges can form only some disjoint stars. The same proof holds for the red edges. Using Lemma 4, we cover the blue edges with three feasible sets, and the red edges with another three feasible sets. So, using at most 6 feasible sets, every edge is covered in the tree. □

It remains to show that the set of positive edges in G can be decomposed into a few number of forests. This can be implied by a direct application of Nash-Williams Theorem (NASH-WILLIAMS 1964). Let G^+ be the subgraph of G with all positive edges. Clearly G^+ is also planar. The Nash-Williams theorem states that the minimum number of forests needed to partition the edges of a graph H is equal to $\max_{S \subseteq V(H)} \frac{m_S}{n_S - 1}$ where m_S, and n_S are the number of edges and vertices in set S respectively. $V(H)$ is the set of all vertices in H. Since G^+ is planar, m_S is at most $3n_S - 6$ for every $S \subseteq V(G^+)$, so three forests are sufficient to cover all edges of G^+. Note that computing these forests can be done in polynomial time using the approach of Gabow and Westerman (Gabow and Westermann 1988). Thus all positive edges in G can be covered with $3 \times 6 = 18$ feasible sets, yielding an 18-approximation algorithm.

Minor Free Graphs

We generalize our results to Minor Free Graphs. A graph H is a minor of G if H is isomorphic to a graph that can be obtained by zero or more edge contractions on a subgraph of G. G is H-Minor Free, if it does not contain H as a minor. A graph is planar iff it has no K_5 or $K_{3,3}$ as a minor (Wagner 1937) (K_5 is a complete graph with 5 vertices, and $K_{3,3}$ is a complete bipartite graph with 3 vertices in each part). We give an $O(h^2 \log h)$-approximation algorithm for OPT-CS-WGG in H-minor free graphs where h

is the number of vertices of graph H. This yields a constant factor approximation for planar graphs in particular, because they are K_5-minor free graphs. We use the following theorem (Thomason 2001). The main property of Minor Free graphs that make them tractable in this problem is sparsity.

Theorem 4. *The number of edges of a H-Minor Free graph G with n vertices is not more than cn where c is equal to $(\alpha + o(1))h\sqrt{\log h}$, h is the number of vertices in H, and α is a constant around 0.319.*

Our planar graph algorithm used *sparsity* to make sure there are low degree vertices at each level and we also need to make sure that the graph is colorable with few colors. Using sparsity we find a proper coloring of these graphs using $2c + 1$ colors. Any subgraph G' of G is also H-Minor Free, having at most $O(c|G'|)$ edges. Thus the average degree of every subgraph of G is at most $2c$, so in every subgraph of G, we can find some vertices with degree at most $2c$ (the average degree). We use this to get a proper $2c+1$-coloring of G. Let v be a vertex in G with degree at most $2c$. $G \setminus \{v\}$ is H-minor free, and inductively we can find a proper $2c + 1$-coloring for it. Vertex v has at most $2c$ neighbors, so one of the $2c + 1$ colors is not used by its neighbors. Thus we can find one appropriate color for v among our $2c + 1$ colors, and get a proper $2c + 1$-coloring for G.

Using theorem 4, every subset S of G has at most $c|S|$ edges, as every subgraph of G is also H-minor free. We then use the Nash-Williams Theorem (NASH-WILLIAMS 1964), and Gabow and Westerman's Algorithm (Gabow and Westermann 1988) to cover all positive edges of G with $O(c)$ forests. We also know that each forest can be decomposed into two unions of stars. The entire graph G is colorable using $2c + 1$ colors, so we need $2c + 1 - 1 = 2c$ feasible sets to cover a union of stars. We conclude that $2c \cdot 2 \cdot O(c) = O(c^2)$ feasible sets are enough to cover all positive edges, and get a $O(c^2)$ approximation algorithm by picking the best (maximum weight) feasible set. Since c is $O(h\sqrt{\log h})$, our approximation factor is $O(h^2 \log h)$.

Bounded Degree Graphs

We now consider bounded degree graphs. A vertex's positive degree is the number of positive edges incident to it. Denote the maximum positive degree in G as Δ. Using the Vizing Theorem, the positive edges can be decomposed into $\Delta+1$ matchings (which are also feasible sets). This yields a $\Delta + 1$ approximation algorithm. A polynomial time method for finding the decomposition can be derived from the Vizing Theorem's proof. We give a *linear time* algorithm for this problem: a randomized $(2+\epsilon)\Delta$ approximation with an expected running time $O(E \log \Delta/\epsilon)$ and $O(V + E)$ space[3] where E is the number of edges.

We pick an arbitrary ordering of the positive edges of the graph, and try to decompose them into $(2 + \epsilon)\Delta$ matchings. We color the edges with $(2 + \epsilon)\Delta$ colors such that no two edges with the same color share an endpoint. Assume that we are in step i, and wish to color the i-th edge in our ordering. Let e be this edge with endpoints u and v. We have already colored $i - 1$ edges and some of those colored edges may have u or v as their endpoints, so we must avoid the color of those edges. For each vertex, we keep the color of its edges in a data structure. Initially the data structures for all vertices are empty. When we color an edge, we add its color to the data structure of its two endpoint vertices. We can use binary search trees to insert and search in $O(\log \Delta)$ time, since we insert at most Δ colors in each data structure. For edge e, we must find a color that is not in the union of the data structures of vertices u and v. There are at most $2(\Delta - 1)$ colors in these two data structures, and there are $(2 + \epsilon)\Delta$ colors in total. Thus if we randomly pick a color, with probability at least $\epsilon/2$, we can use this color for edge e. Checking whether a color is in a data structure can be done using a search query. Thus we can check in time $O(\log \Delta)$ whether the randomly chosen color is good or not. If the color is already taken, we can try again. It takes at most $2/\epsilon$ times in expectation to find an available color. Thus for each edge we spend $O(\log \Delta/\epsilon)$ time to find a color, so the average running time of this decomposition is $O(E \log \Delta/\epsilon)$ in expectation. Finding the best feasible set (matching) does not take more than $O(E)$ time. Thus we get an $(2 + \epsilon)\Delta$-approximation with almost linear running time.

Treewidth Based Approximations

Many hard problems are tractable for graphs with constant or bounded treewidth. We present a polynomial time $O(k^2)$-approximation algorithm where k is the treewidth of the graph, without assuming that the treewidth of graph G is constant or a small number. Algorithms for finding the treewidth of a graph only work in polynomial time when the treewidth is constant. Although we do not know the treewidth, we can still make sure that the approximation factor is not more than $O(k^2)$. We use the following lemma.

Lemma 6. *If G has treewidth k, it has a vertex with degree at most k.*

Proof. Since G has treewidth k, there is a tree T such that every vertex of T has a subset of size at most $k+1$ of vertices of G. The vertices of T that contain a vertex of G form a connected subtree. We also know that the endpoint vertices of each edge in G are in the set of at least one vertex of T together. Now we can prove our claim. Consider a leaf v of T. Let u be the father of v. These two vertices have two subsets of vertices of G like S_u and S_v. If S_v is a subset of S_u, there is no need to keep vertex v in our tree. We can delete it from T, and the remaining tree is also a proper representation of graph G. So we know that there is at least a vertex of G like x which is in S_v, and not in S_u. Clearly v is the only vertex of T that contains x. Otherwise the vertices of T that contain x do not form a connected subgraph. So vertex x can have neighbors only in set S_v which means that x has at most $k + 1 - 1 = k$ neighbors. □

Removing a vertex from a graph does not increase its treewidth, so we can iteratively find vertices of degree at most k, and delete them. Thus we can find a $(k + 1)$ proper

[3]There may be techniques to get rid of the $\log \Delta$ factor in the running time, but resulting in higher space complexity of $O(E + V \cdot \Delta)$ instead of the linear space in our algorithm.

coloring of vertices of G. Using the same decomposition we had for planar graphs, we decompose the positive edges into $O(k)$ matchings and unions of stars. Each matching is a feasible set, and each union of stars can be decomposed into $k+1-1=k$ feasible sets, as we can color the graph with $k+1$ colors. Thus $O(k^2)$ feasible sets are enough to cover all positive edges, yielding an $O(k^2)$ approximation algorithm. Note that we start with a value of k, and we keep deleting vertices with degree at most k. If every vertex is deleted after some number of iterations, we achieve the desired structure. Otherwise at some point the degree of each vertex is greater than k, so the treewidth is more than k. Thus, we can find the minimum k for which every vertex is deleted after some steps, and that k is at most the treewidth of G.

Conclusions and Related Work

Cooperation is a central topic in algorithmic game theory. We studied the optimal coalition structure in WGGs. We showed the problem is $\mathcal{NP}$-hard, but restrictions on the input graph, such as being a tree or having bounded treewidth, result in tractable algorithms. We showed the problem is hard for planar graphs, but provided a polynomial constant approximation algorithm for this class and other classes.

WGGs are a well-known representation of cooperative games, offering a simple way to express synergies. One limitation of our approach is that some games cannot be expressed as a WGG. A general representation of a cooperative game is a table mapping any agent subset to the utility it achieves (with size exponential in the number of the agents). Though the WGG representation is concise for some games, and requires much less space than the exponential size table, some games cannot be expressed as WGGs. Further, even for games given in another representation language and that can be expressed as WGGs, there may not always exist a tractable algorithm for converting the game's representation. To use our methods, one must have the input game given as a WGG. Since the WGG representation is a very prominent representation language for cooperative games, we believe our approach covers many important domains.

Much work was dedicated to team formation, examining representations based on combinatorial structures (Deng and Papadimitriou 1994; Ieong and Shoham 2006; Bachrach et al. 2010; Ohta et al. 2006; Bachrach and Rosenschein 2009) and a survey in (Bilbao 2000). A detailed presentation of such languages is given in (Chalkiadakis, Elkind, and Wooldridge 2011; Shoham and Leyton-Brown 2009). Generating optimal coalition structures received much attention (Shehory and Kraus 1998; Sandholm et al. 1999; Rahwan et al. 2007; Rahwan and Jennings 2008; Service and Adams 2010; Rahwan et al. 2012; **?**) due to its applications, such as vehicle routing and sensor networks. An early approach (Shehory and Kraus 1998) focused on overlapping coalitions, giving a loose approximation.Another approach (Sandholm et al. 1999) has a worst case complexity of $O(n^n)$, whereas dynamic programming approaches (Yun Yeh 1986) have a worst case guarantee of $O(3^n)$. Such algorithms were examined empirically in (Larson and Sandholm 2000). Arguably, the state of the art methods are presented in (Rahwan and Jennings 2008; Rahwan et al. 2012). For general games, such approaches have a worst case runtime of $O(n^n)$ and offers no polynomial runtime guarantees, but can be much faster in practice or under additional assumptions. Such methods assume a black-box that computes the value of a coalition, while we rely on a specific representation. Another approach solves the coalition structure generation problem (Bachrach et al. 2010), but relies on a different representation. A fixed parameter tractable approach was proposed for typed-games (Aziz and de Keijzer 2011) (the running time is exponential in the number of agent "types"). However, in graph games the number of agent types is unbounded, so this is untractable. In contrast to the above, we provide polynomial algorithms and sufficient conditions that guarantee *approximation ratios* for WGGs (Deng and Papadimitriou 1994).

We ignored the issue of coalitional stability. Our methods maximize welfare, but do so in a potentially unstable manner. When agents are selfish, and only care about their own utility, the coalition structure may be broken when some agents decide to form a different coalition, improving their own utility at the expense of others. It would be interesting to examine questions relating to solution concepts such as the core, the nucleolus or the cost of stability(Gillies 1953; Schmeidler 1969; Bachrach et al. 2009b; 2009a; Greco et al. 2011; Chalkiadakis, Markakis, and Jennings 2012).

Several directions remain open for future research. First, as solving the coalition structure generation problem is hard in general WGGs, are there alternative approximation algorithms? Obviously, one can use the algorithms for general games, but we believe a better complexity can be achieved for WGGs than for general games. Second, focusing on the game theoretic motivation for this work, it would be interesting to examine *core-stable* coalition structures rather than just optimal ones. It is not known whether there exists a PTAS[4] for planar graphs or not. Though one might hope to obtain such a PTAS by combining Baker's approach with our algorithm for solving bounded treewitdh graphs, such a direct approach fails. The main reason of this failure is that Baker's approach does not provide a complete partition of the edges in the graph. Baker's technique yields some disjoint cut based subsets of edges such that deletion of each subset results a tractable subgraph, but some of the edges of the graph may not be present in any of these subsets. It turns out that in some hard instances of our problem, edges in the subsets of Baker's approach have large positive weights and the edges outside those subsets have large negative weights. In these cases, removing any of these subsets forces us to have suboptimal (even non-positive) values. If one could find an analogue of Baker's technique that only outputs a complete partition, it may be used in our settings to find some approximation algorithms. Finally, it would be interesting to examine other classes of graphs where one can solve the coalition structure generation in polynomial time. Also, similar tractability results for coalition structure generation could be devised by using other representations.

[4]Polynomial-Time Approximation Scheme.

References

Aumann, R., and Dreze, J. 1974. Cooperative games with coalition structures. *International Journal of Game Theory* 3(4):217–237.

Aziz, H., and de Keijzer, B. 2011. Complexity of coalition structure generation. *Arxiv preprint arXiv:1101.1007*.

Bachrach, Y., and Rosenschein, J. 2009. Power in threshold network flow games. *AAMAS* 18(1):106–132.

Bachrach, Y.; Elkind, E.; Meir, R.; Pasechnik, D.; Zuckerman, M.; Rothe, J.; and Rosenschein, J. 2009a. The cost of stability in coalitional games. *Algorithmic Game Theory* 122–134.

Bachrach, Y.; Meir, R.; Zuckerman, M.; Rothe, J.; and Rosenschein, J. 2009b. The cost of stability in weighted voting games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 1289–1290.

Bachrach, Y.; Meir, R.; Jung, K.; and Kohli, P. 2010. Coalitional Structure Generation in Skill Games. In *AAAI-2010*.

Bansal, N.; Blum, A.; and Chawla, S. 2004. Correlation clustering. *Machine Learning Journal* 86–113.

Bilbao, J. M. 2000. *Cooperative Games on Combinatorial Structures*. Kluwer Publishers.

Chalkiadakis, G.; Elkind, E.; and Wooldridge, M. 2011. *Computational Aspects of Cooperative Game Theory*. Morgan and Claypool.

Chalkiadakis, G.; Markakis, E.; and Jennings, N. R. 2012. Coalitional stability in structured environments. In *AAMAS*, 779–786.

Cowans, P., and Szummer, M. 2005. A graphical model for simultaneous partitioning and labeling. In *AISTATS*.

Deng, X., and Papadimitriou, C. H. 1994. On the complexity of cooperative solution concepts. *Math. Oper. Res.* 19(2):257–266.

Gabow, H., and Westermann, H. 1988. Forests, frames, and games: algorithms for matroid sums and applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, 407–421. New York, NY, USA: ACM.

Gillies, D. B. 1953. *Some theorems on n-person games*. Ph.D. Dissertation, Princeton University.

Greco, G.; Malizia, E.; Palopoli, L.; and Scarcello, F. 2011. On the complexity of the core over coalition structures. In *AAAI*, 216–221.

Håstad, J. 1996. Clique is hard to approximate within $n^{1\text{-epsilon}}$. In *FOCS*, 627–636.

Ieong, S., and Shoham, Y. 2006. Multi-attribute coalitional games. In *ACM EC*.

Larson, K., and Sandholm, T. 2000. Anytime coalition structure generation: average case study. *J. Experimental & Theoretical Artificial Intelligence* 12(1):23–42.

Mitra, P., and Samal, M. 2009. Approximation algorithm for correlation clustering. In *Networked Digital Technologies (NDT)*.

NASH-WILLIAMS, C. S. A. 1964. Decomposition of finite graphs into forests. *J. London Math. Soc.*

Ohta, N.; Iwasaki, A.; Yokoo, M.; Maruono, K.; Conitzer, V.; and Sandholm, T. 2006. A compact representation scheme for coalitional games in open anonymous environments. In *AAAI*, volume 21, 697–702.

Peleg, B., and Sudholter, P. 2007. *Introduction to the theory of cooperative games*. Springer.

Rahwan, T., and Jennings, N. 2008. An improved dynamic programming algorithm for coalition structure generation. In *AAMAS*.

Rahwan, T.; Ramchurn, S.; Dang, V.; Giovannucci, A.; and Jennings, N. 2007. Anytime optimal coalition structure generation. In *AAAI*.

Rahwan, T.; Michalak, T.; Wooldridge, M.; and Jennings, N. R. 2012. Anytime coalition structure generation in multi-agent systems with positive or negative externalities. *Artificial Intelligence*.

Sandholm, T.; Larson, K.; Andersson, M.; Shehory, O.; and Tohmé, F. 1999. Coalition structure generation with worst case guarantees. *AIJ*.

Schmeidler, D. 1969. The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics* 17(6):1163–1170.

Service, T., and Adams, J. 2010. Approximate coalition structure generation. In *AAAI*.

Shehory, O., and Kraus, S. 1998. Methods for task allocation via agent coalition formation. *Artificial Intelligence* 101(1–2):165–200.

Shoham, Y., and Leyton-Brown, K. 2009. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge Univ Pr.

Tan, J. 2008. A note on the inapproximability of correlation clustering. *Information Processing Letters* 108(5):331–335.

Thomason, A. 2001. The extremal function for complete minors. *J. Comb. Theory Ser. B* 81(2):318–338.

Voice, T.; Polukarov, M.; and Jennings, N. R. 2012. Graph coalition structure generation. Technical report.

Wagner, K. 1937. Uber eine Eigenschaft der ebenen Komplexe. *Math. Ann.* 114:570–590.

Yun Yeh, D. 1986. A dynamic programming approach to the complete set partitioning problem. *BIT Numerical Mathematics* 26(4):467–474.