Anchored Team Formation Games

Jacob Schlueter,[1] Christian Addington,[1] Judy Goldsmith[1]
University of Kentucky
Lexington, Kentucky
tjacobschlueter@gmail.com, crad225@uky.edu, goldsmit@cs.uky.edu

Abstract

We propose Anchored Team Formation Games (ATFGs), a new class of hedonic game inspired by tabletop role playing games. We establish the NP-hardness of determining whether Nash stable coalition structures exist, and provide results for three heuristics for this problem. We highlight costs and benefits of each heuristic and provide evidence that all three are capable of finding Nash stable coalition structures, when they exist, much more quickly than a deterministic algorithm.

Introduction

Consider tabletop role playing games (TRPG) such as Dungeons and Dragons. In order to play a TRPG, a group must have players and a *game manager (GM)*; the former working through challenges set up by the latter. The expectation of enjoying a particular group to play such a game is based on expectations that the GM will set up a good story line, with sufficient challenges and rewards, and that the fellow travelers will offer good measures of cooperation and competition. We introduce *anchored team formation games (ATFGs)* to investigate the formation of groups with a leader, or anchor, using the formation of TRPG groups as an example application. While we refer to the gaming application throughout the paper, the anchor could be a team lead in a programming or engineering team, in business, class project groups, or outdoor adventuring. Whatever the application, the driving question is, how do we divide individuals into groups that will get the job (or game) done, and choose leaders for the groups, in a way that is consistent with the individuals' preferences?

We are interested in determining whether *stable coalition structures (stable CSs)* exist for a given ATFG, and how to find them. Given our proposed use case, we believe Nash stability is the most relevant stability notion in the literature. However, the problem of finding a Nash stable CS is NP-hard. We present experimental results with three effective heuristics to check the existence of Nash stable coalition structures in ATFGs. Our first algorithm starts by selecting anchors around which to build coalitions, then divide players in a round robin fashion consisting of several rounds where each coalition chooses one player to add. Our second algorithm is a local search implementation that attempts to minimize the number of blocking players, those who can improve their utility by unilaterally deviating from their assignment. Our third algorithm daisy-chains the previous two, using the output of the round-robin algorithm as a starting place for a local search.

Experiments on 10–12-agent instances show that the first two algorithms perform sufficiently well that we did not need to test the third. On larger instances, we see that daisy-chaining is a significant improvement over either individual algorithm. These experiments indicate that good coalition structures can be found.

In the next section, we give a brief survey of related work. We formally define hedonic games, ATFGs, and Nash stability, then present the algorithms used, the experimental set-up, and results.

Related Work

Our work is in the area of hedonic coalition formation games, often shortened to hedonic games. Hedonic games are a broad category of coalition formation games, formally codified by Banerjee, Konishi, and Sönmez and Bogomolnaia and Jackson, where each player's utility is wholly derived from their coalition and is non-transferable [4, 5]. General case hedonic games can model a wide variety of problems, but it is difficult to generalize about the computational complexity of determining the existence, or finding, stable or optimal coalition structures over all hedonic games. However, there are some results for general hedonic games, which provide upper bounds on the complexity for all games they generalize [3]. Much hedonic games research focuses on subclasses that exhibit certain useful properties. Early examples, which predate the formal codification of hedonic games, are Gale and Shapley's stable marriage and stable roommates problems, which always have polynomially computable stable matchings [8]. In both problems players rank each other based on whom they want to be paired with, which restricts their scope and applicability; stable marriage further restricts rankings to members of the opposite gender [8] or other set in some bipartite graph (e.g., medical resi-

---

dents and hospitals [17]).

Banerjee, Konishi, and Sönmez and Bogomolnaia and Jackson introduced additively separable hedonic games (ASHGs), where players assign utility values to each other, similar to the rankings in Gale and Shapley's roommates problem, but there are no restrictions on coalition size [4, 5, 8]. ASHGs are of particular interest to our work, because they are generalized by ATFGs; there are many other hedonic games inspired by ASHGs [1, 6, 13, 14, 20]. There are many hardness results for ASHGs and their variants, which establish some baselines for our work [2, 15, 16, 19, 29].

Our work on ATFGs expands the body of hedonic games inspired by gaming. Hedonic games inspired by gaming include *Tiered Coalition Formation Games (TCFGs)* inspired by *Pokémon*, *Roles and Teams Hedonic Games (RTHGs)* inspired by League of Legends, and *Role Based Hedonic Games (RBHGs)*, which generalize RTHGs [21, 22, 23].

We expand the body of research applying heuristics to hedonic games with our work on ATFGs. Spradling et al. gives experimental results with greedy heuristics for RTHGs [23]. Later work by Spradling applies greedy heuristics to RBHGs [24]. Keinänen and Keinänen et al. develop local search heuristics for core stability verification and social welfare optimization in hedonic games [9, 10, 12]. We conjecture that Keinänen et al. does not extend this to local search algorithms to construct stable CSs, as we do, in part because the fitness function we use is unusual, and is specific to Nash stability. However, Keinänen presents an EXPTIME breadth first search algorithm to compute all Nash stable CS in ASHGs [11], which they were able to run on (many) instances of size 10. Our complete algorithm has successfully run 20-agent instances, despite ATFGs being more complicated the ASHGs. Waxman, Kraus, and Hazon utilize simulated annealing and leximin heuristics to optimize egalitarian welfare in ASHGs where a fixed number of coalitions must be formed [28]. Taywade, Goldsmith, and Harrison give experimental results with three heuristics to optimize social welfare in decentralized matching [27], which is significantly different from our centralized setting, as well as being focused on optimality rather than stability.

Preliminaries

ATFGs are a subclass of hedonic games, so we establish a formal definition of this parent class of games.

Definition 1. *[4, 5] **Hedonic games** are coalition formation games with nontransferable utility wherein players' preferences are concerned only with their own coalition. This inherently self-interested means of determining utility makes such games hedonic in nature.*

*Let $\mathcal{N}_i$ be the set of possible coalitions containing player $i \in$ player set N. A preference ordering of $\mathcal{N}_i$ is derived from the preference set $P_i \in$ the set of all preferences P. A solution for a game is a partition, called a **coalition structure (CS)** γ; the set of all CSs is Γ. Each player $i \in N$ has preferences over all $\gamma \in \Gamma$ based solely on their assigned coalition in each γ. We use $u_i(\gamma)$ as shorthand for $u_i(C)$, the utility player $i \in N$ derives from coalition $C \in \gamma$ such that $i \in C$.*

ASHGs are a popular subclass of hedonic games relevant to our ATFG results.

Definition 2. *[4] **Additively Separable Hedonic Games (ASHG)** are a class of hedonic game where each player $i \in N$ assigns values to each player $j \in N$, expressed as $v_i(j)$; for all i, $v_i(i)$ is 0. The utility a player a_i derives from each coalition $S \in \mathcal{N}_i$ such that $i \in S$ is defined as $u_i(S) = \sum_{j \in S} v_i(j)$.*

We now introduce our model of team formation for gaming. A standard play group in many TRPGs consists of four players and a fifth person, called the *Game Manager (GM)*. The GM provides the game's setting and leads the players through the story. The number of players varies, but there is only one GM. A GM's performance can easily make or break a game. A good GM keeps players engaged, while a bad GM can make for a poor experience. Since many TRPG groups are composed of people who are well-acquainted with each other, we propose the notion of pair utility values, reflecting the positive or negative synergy a given pair may bring to the group; a pair of friends may role-play well together and help keep the game focused, or they could distract each other by making inside jokes.

We introduce Anchored Team Formation Games to model the formation of such groups. One of the biggest differences between our setting and most others is that we assume that players may have opinions not only on other individuals, but also on *pairs* of players, either positively[1] or negatively.

Anchored Team Formation Games are a class of cooperative coalition formation games in which each coalition must contain an anchor, or leading player, and in which upper and lower bounds limit the permissible sizes of coalitions.

Definition 3. *An **Anchored Team Formation Game (ATFG)** is a tuple $\langle N, V, P, D, c_u, c_l \rangle$, where N is a set of n players, and V, P, and D define weight values such that each $i \in N$, there are vectors v_i, p_i, and d_i of lengths n, n^2, and n respectively. The $v_i[j]$ is the utility i gets for being in a coalition with player $j \in N$; $p_i[j,k]$ is the utility i gets for being in a coalition with the pair of players (j,k); $d_i[j]$ is the utility i gets if j is the anchor (GM) for i's coalition. Values contained in each v_i, p_i, and d_i are assumed to either be integers or unknowns.*

The values c_u and c_l define upper and lower bounds on coalition size. Any valid coalition $C \subseteq N$ must satisfy $c_u \geq |C| \geq c_l$. Further, all valid coalitions $C \subseteq N$ must contain a designated player $g(C)$ who serves as the anchor, or coalition leader. In order for a CS γ to be valid, it must consist solely of valid disjoint coalitions.

The utility $u_i(\gamma)$ a player $i \in N$ derives from a valid CS γ is defined as follows:

$$d_i[g(C_i)] + \sum_{j \in C_i \setminus \{i\}} v_i[j] + \sum_{\{j,l\} \subset C_i \setminus \{i\}} p_i[\{j,l\}].$$

Observation 1. *Evaluating $u_i(C)$ takes time $O(n^2)$.*

---

[1] Consider James and Elyse Willems (https://www.youtube.com/channel/UCboMX_UNgaPBsUOIgasn3-Q): most players would yield a higher utility from having both James and Elyse in a TRPG campaign as opposed to either, singly.

Note that evaluating $u_i(C)$ takes time $O(m^2) \subseteq O(n^2)$, where m is an upper bound on $|C|$, due to the p_i summation. Thus, evaluating the total utility of a coalition is $O(m^2)$ (and thus $O(n^2)$) and of an entire CS is $O(n^3)$, or more precisely, $O(nm^2)$.

Stability

An assignment of players to tables is only useful if the players consent to the assignment; we presume that, if they are aware of an assignment that Pareto-dominates the one on offer, they will move to the better assignment. Pareto dominance is one of many forms of *stability*, the idea that a CS will not be disrupted by players rejecting their assigned coalitions and moving to other coalitions. There are many sets of constraints placed on such disruptions, such as the number of players that can move simultaneously; whether all moving players must see an increase in utility; whether players left behind by movers must see their utility increase, or whether players being joined by movers must see their utility improve. We focus on Nash stability, which was adapted to hedonic games by Bogomolnaia and Jackson [5].

Definition 4. *[5] A CS is **Nash stable** for a coalition formation game if no individual player can improve their utility by deviating from their current coalition to join another coalition or to become a singleton.*

Note that there are some trivial cases of stable CSs, which we ignore. For instance, if coalitions must have size ≥ 3, then the CS of all singletons is Nash stable, since no pair is a valid coalition. We assume that all CSs contain at least one valid coalition.

Our local search heuristic (Algorithm 2) defines fitness by the number of blocking players, players who want to unilaterally leave their current coalition to join another or to become a singleton. This measure, the *Degree of Instability (DoI)*, was suggested by Roth and Xing for matching in decentralized markets [18]. A more robust notion of instability based on blocking *pairs* was proposed by Eriksson and Häggström [7]. We use the simpler count of blocking players as it is more appropriate for Nash stability.

Complexity of ATFGs

Proposition 1. *Nash stability verification for ATFGs runs in polynomial time.*

Proof. Consider some ATFG (N, V, P, D, c_u, c_l) with CS $\gamma = \{C_1, ..., C_k\}$. For each $i \in N$, recall that $u_i(\gamma)$ is defined by:

$$d_i[g(C_i)] + \sum_{j \in C_i \setminus \{i\}} v_i[j] + \sum_{\{j,l\} \subset C_i \setminus \{i\}} p_i[\{j, l\}].$$

For each i and each $C' \in \gamma$, we compute $u_i(C' \cup \{i\})$ in time $O(n^2)$, by Observation 1. If $\exists i : u_i(C' \cup \{i\}) > u_i(\gamma)$, the CS is not Nash stable. This computation is order $O(n^4)$, as there are n players and $O(n)$ coalitions. □

Proposition 2 highlights the relationship between ATFGs and ASHGs.

Proposition 2. *ATFGs generalize ASHGs.*

Proof. We can convert any ASHG into an ATFG as follows:

1. Taking the values each player assigns to each other player as-is.
2. For all $p_i \in P$ set all values $p \in p_i$ to 0.
3. For all $d_i \in D$ set all values $d \in d_i$ to 0.
4. Set $c_u = n$.
5. Set $c_l = 0$.

Following this conversion, the anchor becomes irrelevant, as they have no impact on utility. The bounds set on coalitions are such that they impose no limits whatsoever, as any coalition size obtainable from a set of n players is permitted. □

Sung and Dimitrov prove that determining if a Nash stable CS exists is NP-complete for ASHGs [25]. Their proof shows that this holds even when there is a bound of 7 on the size of coalitions. Because ATFGs generalize ASHGs, we know that the Nash stability existence problem is NP-hard for ATFGs. Proposition 1 shows that Nash stability verification is in P, so Nash stability existence is in NP. Thus Nash stability existence is NP-complete.

An ATFG constructed from an ASHG does not make use of coalition size restrictions or utility values assigned to anchors or pairs of players. Our focus is on settings where coalition sizes are restricted and a coalition's anchor is an important part of each player's utility.

Algorithms

For the heuristic algorithms, the first step is to choose the GMs, and then to assign players to coalitions. We present the GM selection function, and then describe how agents' utilities are represented. We then describe local search and round-robin algorithms, and the complete algorithm.

In all algorithms, W represents an arbitrary ATFG instance (N, V, P, D, c_u, c_l).

Definition 5. *We use a **GM Selection** algorithm to select agents who are willing to be GMs. The algorithm checks the value each agent derives from being a GM, then places all agents that derive positive value from being a GM into a list of potential GMs. Next, a number of GMs, $g_s \leftarrow s \in [\lceil \frac{|N|}{c_u} \rceil, \lfloor \frac{|N|}{c_l} \rfloor]$, are selected at random from that list. As GMs are selected, they are moved to the final GM list. The list of remaining agents becomes the list of players.*

Note that the GM Selection algorithm runs in time $O(n)$, assuming that the random number generation takes constant time.

Next, we introduce introduce several algorithms to assign these players to tables.

Complete search

As a baseline for our heuristics, we have a branch and bound algorithm that finds all Nash stable partitions of a given instance. We refer to this as the *complete algorithm.* It is based on a depth-first search tree, where nodes at each level consist of coalitions that can be constructed from as-yet unassigned agents. To eliminate redundant branches, each node in the search tree contains the lowest-numbered available node.

Branches are pruned if a node introduces instability, either because agents in the new coalition prefer to join an earlier coalition, or agents in earlier coalitions prefer to join the new one. Furthermore, singleton coalitions are disallowed as nodes, branches are pruned if they cannot be extended stably without singletons.

We ran the complete algorithm on 20 instances with 15 agents and 25 instances with 20 agents. The 15 agent instances averaged 22.7 seconds of wall clock runtime with a standard error of 3.8 seconds on our virtual machine, which was a Linux machine with a CPU of Intel Xeon 2 cores at 2.1GHz, and 4GB of RAM. The 20 agent instances averaged 6489.5 seconds of runtime with a standard error of 1270.4 seconds on the same machine. We started three 24 agent instances to completion, but all three were terminated after 24 or more hours.

Heuristics

Our first heuristic is a greedy round robin.

Definition 6. *The **Utilitarian Round Robin (URR)** algorithm first chooses GMs. The GMs then take turns choosing players whose addition to the table maximizes total utility for the table, this utility value is computed by Algorithm 1.*

Algorithm 1 Update coalition values

```
Input: ATFG $W$, GM $g$, player set $A^*$, Coalition $C$ s.t. $g \in C$
Output: $M_g$ % Pairs $\langle i, m_i \rangle$ where $m_i$ is the marginal utility of adding $i$ to $C$
$M_g \leftarrow \emptyset$
$util_C \leftarrow 0$
for $k \in C$ do
    $util_C \leftarrow util_C + u_k(C)$
end for
for $i \in A^*$ do
    $util \leftarrow 0$
    for $k \in C \cup \{i\}$ do
        $util \leftarrow util + u_k(C \cup \{i\})$
    end for
    $m_i \leftarrow util - util_C$
    $M_g \leftarrow M_g \cup \{\langle i, m_i \rangle\}$
end for
return: $M_g$
```

The URR algorithm maintains a matrix of valuations of players by GMs, representing the value of adding a player to that GM's table. Choosing a best addition is $O(n)$ for each of the $O(n)$ GMs. After each round of additions, the $O(n^2)$ matrix is updated, with each update taking $O(n^2)$, by Observation 1. There are $O(n)$ rounds, so an iteration runs in time $O(n^5)$.

Algorithm 2 is a local search heuristic, with the goal of minimizing the number of blocking players. Each iteration of the algorithm chooses a random set of players to be GMs, and randomly assigns other players to the GMs' tables. It then repeatedly chooses a neighboring CS (defined by a single player being assigned to a different table) that improves (lowers) the fitness. Note that it is possible that a CS is not Nash stable, but has no improving neighbor — even if there is a Nash stable CS for that ATFG instance. This local-but-not-global optimum is a typical phenomenon of local search algorithms; to handle this, we use multiple random re-starts.

An iteration of local search has $O(n)$ improvements of fitness, since at most n agents can *want* to deviate at a given time. Further, since an agent only successfully moves if the movement improves the fitness, we avoid potential loops. The algorithm maintains a GM by players matrix of valuations, indicating the value each player derives from each GM's table. A scan of a player's row can identify if they wish to move; to test stability requires scanning the $O(n^2)$ matrix, and potentially updating it if a player moves. Each move of player i from GM j to GM k requires $O(n^2)$-time updates to the utilities of all $O(n)$ players for the coalitions with GMs j and k. The hardest part of the algorithm is determining which agent, if any, want to move. For each of the n potential movers, i, and for each of the $O(n)$ coalitions C they might join, we check, for each other agent j, does j newly want to move (to $C \cup \{i\}$, to the coalition i just left, or another coalition if $j \in C$ and j doesn't like the addition of i)? Determining if j wants to move takes $O(n^2)$, by Observation 1. Thus, the entire check for acceptable moves takes $O(n^5)$, and an iteration of LS, involving $O(n)$ such checks, is $O(n^6)$.

Algorithm 2 Local Search

```
Input: ATFG $W$, GM set $G$, player set $A^*$
Output: $\gamma$, Potentially Nash stable CS
$\gamma \leftarrow \emptyset$ %randomly assigned CS
for all $g \in G$ do
    $\gamma_g \leftarrow \frac{|A^*|}{|G|}$ random players $^+$
    $\gamma \leftarrow \gamma \cup \{\{g\} \cup \gamma_g\}$
end for
$M \leftarrow \emptyset$
for all $i \in N$ do
    if $i$ can receive higher utility by moving then
        $m_i \leftarrow \#$ players wanting to move after $i$ moves
        $M \leftarrow M \cup \{m_i\}$
    end if
end for
while $\exists m \in M : m < |M|$ do
    choose $\{i : m_i = \min(M)\}$
    $\gamma \leftarrow \gamma$ modified by $i$'s move
    $M \leftarrow \emptyset$
    for all $i \in N$ do
        if $i$ can receive higher utility by moving then
            $m_i \leftarrow \#$ players wanting to move after $i$ moves
            $M \leftarrow M \cup \{m_i\}$
        end if
    end for
end while
return: $\gamma$
```

$^+$ $\frac{|A^*|}{|G|}$ is $\left\lceil \frac{|A^*|}{|G|} \right\rceil$ or $\left\lfloor \frac{|A^*|}{|G|} \right\rfloor$ based on players available

Our third heuristic, *DC,* daisy-chains URR and local search, by using the CS output by URR to hot-start the local search algorithm. The complexity of DC is d $O(n^6)$.

Experiments

Our experimental contributions are results for our three greedy heuristic algorithms, demonstrating the relative benefits and detriments of each and showing that all three outperform a deterministic algorithm on instances with 12 or more players.

Experimental Setup

Our heuristics were coded in Python 3.7 and currently do not rely on third-party packages. We tested our heuristics against four hand-crafted benchmark instances with known Nash stable CSs, with 10, 12, and 25 players, respectively. We also tested the heuristics against randomly generated instances; 50 with 10 players; 20 with 12 players; 25 with 20 players; 25 with 24 players; 20 with 25 players; 25 with 30 players. To randomly generate these instances, we used coalition size limits of 3–5 for 10, 12, and 15 player instances and 3–8 for 20, 24, 25, and 30 player instances, asymmetric valuations between $-n$ and n inclusive for individuals, pairs, and GMs, and a pair valuation probability of 15%.

Using the complete algorithm, we found Nash stable CSs for 24 of the randomly generated 10-player instances and 12 of the 12-player instances. We then ran the heuristic algorithms with fixed maximum numbers of random restarts. (Note that the heuristic algorithms halt as soon as they find a single Nash stable CS for the given instance.)

For each of our heuristics, we conducted 10 trials each for the 10, 12, 15, 20, 24, 25, and 30 player instances. Trials for utilitarian round robin heuristic were capped at 100,000 restarts, while trials for local search and daisy-chain were capped at 10,000 restarts. Our experiments were run on Linux Ubuntu with an Intel Xeon Processor, with 2.1GHz CPU and 4GB of RAM. We collected data on the total number of restarts before a Nash stable CS was found, and the time duration of the trials, as well as the number of restarts and time per size of instance.

Results

We observe that local search is significantly slower per iteration than URR, and thus our tests of LS run slower than those of URR, even using a factor of 10 fewer restarts. However, running the two algorithms in a daisy chain is much faster than running local search on its own.

Local search also does poorly in finding stable CSs. The number of possible CSs is *much* larger than the number of Nash stable CSs for most of these instances, so LS has to be extremely lucky to be started near a local optimum that is in fact globally optimal.

While URR was not as effective as we had hoped in finding Nash stable CSs, we believe it often provided CSs that were close to stable, i.e., with *low degree of instability,* meaning, here the number of agents that would individually prefer to deviate from their assigned coalitions. Using local search to relocate those individuals worked significantly better than either algorithm individually. (See Table 2.)

We observe that the percentage of instances for which the heuristics found Nash stable CSs is quite small for instances of sizes 25 and 30. We attribute this to an insufficiency of random restarts; with a larger number of cores on our VMs, we are confident that the algorithms will run significantly faster, and these percentages will increase. We were unable (so far) to verify the existence of Nash stable CSs on those instances using the complete algorithm, but with high probability, a larger percentage of them have Nash stable CSs than these results indicate.

Table 1: Average run time (CPU seconds) and percentage of instances for which NS coalitions are found.

# Players	URR		LS		DC	
	time	%	time	%	time	%
10	24.5	47.3	16.2	**47.9**	2.0	36.5
12	43.2	17	52.9	17.5	22.9	**52.5**
15	239.0	65.5	224.5	0.5	44.6	**71.5**
20	898.4	20.4	328.6	0	178.1	**25**
24	268.4	**94.4**	588.4	0	748.5	79.6
25	1745.7	0	1156.4	0	591.9	**4**
30	2605.8	**7.6**	1936.2	0	1533.8	2.5

Note: For the 24 agent DC test, 50,000 restarts were allowed.

Of the randomly generated instances: 48% of our 10 agent instances are stabilizable, 60% of 12 agent instances are stabilizable, 95% of 15 agent instances are stabilizable, and 96% of 20 agent instances are stabilizable. We conjecture that a smaller percentage of 10 and 12 agent instances are stabilizable because they both have utility ranges from -20 to +20, whereas the other instances were generated with limits no wider than -n to $+n$ where n is the number of agents.

We provide information on the degree of instability (DoI), the number of agents who prefer to move to show the improvements from the URR hot start.

Table 2: Average DoI

# Players	Number of movers		
	URR	LS	DC
15	0.345	0.885	**0.045**
20	0.952	2.676	**0.116**
24	0.056	4.284	**0.028**
25	2.455	4.36	**1.105**
30	1.3	6.892	**1.18**

Conclusions and Future Work

We have introduced a novel coalition formation game for a subject dear to our hearts, namely social and (in pre-pandemic days) in person gaming. Like many such cooperative games, finding stable CS is NP-hard. However, we have introduced three fast and effective heuristic algorithms to find Nash stable CS. We focus on Nash stability because we believe it is, socially, the most relevant notion in the literature; though we can imagine social cliques that would be blocking coalitions for core stability, we imagine that such cliques would not throw themselves into the centralized assignment process.

Future work may include decentralized matching algorithms for ATFGs in the style of Taywade et al. [26, 27].

References

[1] Aziz, H.; Brandt, F.; and Harrenstein, P. 2014. Fractional hedonic games. In *AAMAS*, 5–12. International Foundation for Autonomous Agents and Multiagent Systems.

[2] Aziz, H.; Brandt, F.; and Seedig, H. G. 2013. Computing desirable partitions in additively separable hedonic games. *Artificial Intelligence* 195: 316 – 334.

[3] Ballester, C. 2004. NP-completeness in hedonic games. *Games and Economic Behavior* 49(1): 1 – 30.

[4] Banerjee, S.; Konishi, H.; and Sönmez, T. 2001. Core in a simple coalition formation game. *Social Choice and Welfare* 18(1): 135–153.

[5] Bogomolnaia, A.; and Jackson, M. O. 2002. The stability of hedonic coalition structures. *Games and Economic Behavior* 38(2): 201–230.

[6] Dimitrov, D.; Borm, P.; Hendrickx, R.; and Sung, S. C. 2006. Simple priorities and core stability in hedonic games. *Social Choice and Welfare* 26(2): 421–433.

[7] Eriksson, K.; and Häggström, O. 2008. Instability of matchings in decentralized markets with various preference structures. *International Journal of Game Theory* 36(3-4): 409–420.

[8] Gale, D.; and Shapley, L. S. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly* 69(1): 9–15.

[9] Keinänen, H. 2009. Local Search Algorithms for Core Checking in Hedonic Coalition Games. In *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems: First International Conference, ICCCI 2009, Wroclaw, Poland, October 5-7, 2009, Proceedings*, volume 5796, 51. Springer Science & Business Media.

[10] Keinänen, H. 2009. Simulated annealing for multiagent coalition formation. In *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, 30–39. Springer.

[11] Keinänen, H. 2010. An algorithm for generating Nash stable coalition structures in hedonic games. In *International Symposium on Foundations of Information and Knowledge Systems*, 25–39. Springer.

[12] Keinänen, H.; et al. 2011. *Algorithms for coalitional games*. Master's thesis, Turku School of Economics Ae-2: 2011.

[13] Lang, J.; Rey, A.; Rothe, J.; Schadrack, H.; and Schend, L. 2015. Representing and solving hedonic games with ordinal preferences and thresholds. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 1229–1237. International Foundation for Autonomous Agents and Multiagent Systems.

[14] Nguyen, N.-T.; Rey, A.; Rey, L.; Rothe, J.; and Schend, L. 2016. Altruistic hedonic games. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 251–259. International Foundation for Autonomous Agents and Multiagent Systems.

[15] Peters, D. 2017. Precise complexity of the core in dichotomous and additive hedonic games. In *International Conference on Algorithmic DecisionTheory*, 214–227. Springer.

[16] Rey, A.; Rothe, J.; Schadrack, H.; and Schend, L. 2016. Toward the complexity of the existence of wonderfully stable partitions and strictly core stable coalition structures in enemy-oriented hedonic games. *Annals of Mathematics and Artificial Intelligence* 77(3): 317–333.

[17] Roth, A. E. 1984. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of political Economy* 92(6): 991–1016.

[18] Roth, A. E.; and Xing, X. 1997. Turnaround time and bottlenecks in market clearing: Decentralized matching in the market for clinical psychologists. *Journal of political Economy* 105(2): 284–329.

[19] Schlueter, J.; and Goldsmith, J. 2020. Internal Stability in Hedonic Games. In *FLAIRS*.

[20] Schlueter, J.; and Goldsmith, J. 2020. Super Altruistic Hedonic Games. In *FLAIRS*.

[21] Siler, C. 2017. Tiered Coalition Formation Games. In *The Thirtieth International Flairs Conference*.

[22] Spradling, M.; and Goldsmith, J. 2015. Stability in role based hedonic games. In *The Twenty-Eighth International FLAIRS Conference*.

[23] Spradling, M.; Goldsmith, J.; Liu, X.; Dadi, C.; and Li, Z. 2013. Roles and teams hedonic game. In *International Conference on Algorithmic Decision Theory*, 351–362. Springer.

[24] Spradling, M. J. 2017. Optimizing Expected Utility and Stability in Role Based Hedonic Games. In *The Thirtieth International Flairs Conference*.

[25] Sung, S.-C.; and Dimitrov, D. 2010. Computational complexity in additive hedonic games. *European Journal of Operational Research* 203(3): 635–639.

[26] Taywade, K.; Goldsmith, J.; and Harrison, B. 2018. Decentralized Multiagent Approach for Hedonic Games. In *16th European Conference on Multi-Agent Systems*.

[27] Taywade, K.; Goldsmith, J.; and Harrison, B. 2020. Decentralized Marriage Models. In *The Thirty-Third International Flairs Conference*.

[28] Waxman, N.; Kraus, S.; and Hazon, N. 2020. On Maximizing Egalitarian Value in K-coalitional Hedonic Games. *arXiv preprint arXiv:2001.10772* .

[29] Woeginger, G. J. 2013. Core Stability in Hedonic Coalition Formation. In *International Conference on Current Trends in Theory and Practice of Computer Science*, 33–50. Springer.