2015

Role Based Hedonic Games

Matthew Spradling
University of Kentucky, mjspra@umflint.edu

Role Based Hedonic Games

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy
in the College of Engineering at the
University of Kentucky

By
Matthew Spradling
Lexington, Kentucky

Director: Dr. Judy Goldsmith, Professor of Computer Science
Lexington, Kentucky 2015

ABSTRACT OF DISSERTATION

Role Based Hedonic Games

In the hedonic coalition formation game model Roles Based Hedonic Games (RBHG), agents view teams as compositions of available roles. An agent's utility for a partition is based upon which role she fulfills within the coalition and which additional roles are being fulfilled within the coalition. I consider optimization and stability problems for settings with variable power on the part of the central authority and on the part of the agents. I prove several of these problems to be NP-complete or coNP-complete. I introduce heuristic methods for approximating solutions for a variety of these hard problems. I validate heuristics on real-world data scraped from League of Legends games.

KEYWORDS: coalition formation, computational complexity, hedonic games, optimization

Author's signature: Matthew Spradling

Date: August 4, 2015

Role Based Hedonic Games

By
Matthew Spradling

Director of Dissertation: Judy Goldsmith

Director of Graduate Studies: Miroslaw Truszczyński

Date: August 4, 2015

To my brother. John, I am always on your team.

ACKNOWLEDGMENTS

I honor the memory of Duncan Clarke, and thank Martha Clarke for her generous support. I hope that in my career I may earn and at least attempt to repay this in kind to the next student in need of a hand.

Thank you to Dr. De Liu for your wonderful advice on the progress of my work, in particular for your suggestion to consider stable assignments in role based hedonic games. Your thoughts have opened up doors to many important problems I otherwise may not have considered.

Thank you to Dr. Andy Klapper for impressing upon me the need to do my best work and to seek the same of others. Dr. Grzegorz Wasilkowski, thank you for listening when I had trouble. Both of you changed my life.

Thank you to Dr. Jinze Liu and Dr. Victor Marek for introducing me to databases and data-mining. To Dr. Liu, thank you for allowing me the opportunity to pursue my research interests as a part of your data-mining course. To Dr. Marek, thank you for giving me the opportunity to really push myself with database programming. I owe my knowledge and abilities here to both of you.

Thank you to Dr. Jerzy Jaromczyk and Dr. Kenneth Calvert for giving me the chance to teach courses alone during the summer! You may not be aware, but this made me the last person to teach CS 115 in c++ at UK. :) Thank you for involving me in the conversations about transitioning to Python–the switch has made a world of difference for the students. I would not have developed my confidence as a teacher, in and out of the classroom, to nearly the same degree without these opportunities.

Thank you to Professor Paul Piwowarski for teaching me the fundamentals, the skill of transitioning from language to language with ease, for granting me the great opportunity to develop an educational game for my senior design project, and in general for teaching me such good practices in testing and documentation of code. I owe much of my professionalism as a programmer to your mentoring.

Thank you to Dr. Mirek Truszczynski for meeting with me even when I was an undergraduate, for listening when I was unclear, and always inviting me back again. Thank you, also, for not being too upset the day I laughed far too loudly in the Marksbury building.

Thank you to Dr. Judy Goldsmith for introducing me to Dr. Truszczynski and so many other important people, places and things. Thank you for putting the thought in my head that graduate school wasn't a pipe dream. Thank you for helping me through some of the most terrifying and wonderful times of my life thus far. Thank you for all of your advice, and for listening when I really needed it.

Thank you to Dr. Debby Keen for making all of the above possible. Thank you for taking an interest when I was doing free tutoring in RGAN. The trajectory of my life, at UK and beyond, was shifted entirely because you gave me the opportunity to join up as an undergraduate assistant. Your impact on my life has never stopped since then. While I can almost certainly never repay you, I sincerely hope that life will do so. I will still try.

Dad, thank you for working so hard and always finding a way to make things work. I have at least some idea now of how terrifying it can be. I am so happy that you and mom will be able to retire soon. Mom, I don't understand how, but you seem to be capable of accomplishing anything you decide is worth your while to do. I will strive for that. Thank you for surrounding me with opportunities to learn in every way that you could.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introduction

In this chapter, I provide real-world motivation for a team formation model focused upon hedonic utilities for *roles* and *compositions of roles* within teams. I introduce this as the *role based hedonic game* (RBHG) model, which is formally defined in Section 2.2.

I detail motivations in *massively multiplayer online games* (MMOs), urban and planetary robot exploration, and modular robot teams in Section 1.1. Following in Section 1.2, I describe the stabilization and optimization goals of matchmaking in the RBHG setting and give a brief overview of the results provided in this dissertation. I provide an outline of the remaining chapters in Section 1.3.

1.1 A Massively Multiplayer Motivation

In coalition formation games, agents from a population have various utilities for different partitions. Hedonic coalition formation games are a sub-class in which agents have utilities for their own teams and not others. In this dissertation, I consider the variant *Role Based Hedonic Games* (RBHGs) [68]. In this model, an agent's utility for a partition is based upon the *role* it fulfills on its team and the roles fulfilled by its teammates. The multiset of roles fulfilled by a team is termed its *composition*. Team formation includes forming a partition of agents to teams and matching agents to roles within their teams.

This work is motivated by team formation in massively multiplayer online games. World of Warcraft [34], League of Legends [44], Defense of the Ancients 2 (DoTA 2) [66], Counter–Strike: Global Offensive [65] and Diablo III [32] are currently the five most popular online games in the world, taking up over 44% of the share of playing time according to the February 2015 report by Raptr [58].

While these games vary highly in play style, each incorporates some variation of role based team formation. Players may fulfill different roles within teams depending upon play styles, goals and personal preferences. Roles within a team may be defined loosely, as is the case with Counter-Strike, or more strictly as with *multiplayer online battle arenas* (MOBAs) such as League of Legends and DoTA 2. In a typical MOBA, players join a queue and are partitioned into teams of a fixed size to compete team–versus–team. Within a team, each player fulfills a single role such as *attacker*, *defender* or *support*. Game play involves attacking the enemy team and the enemy "base" until achieving some victory objective. League of Legends is currently the most popular game in the MOBA genre and in general, taking up over 21% of the share of playing time on its own [58]. Other popular entries in the MOBA genre include Heroes of Newerth [71] and *SMITE* [45]. The most recent entry into the MOBA lineup is Heroes of the Storm [33], released by Blizzard Entertainment in June, 2015. With several major developers investing in this popular genre of video game, I consider the importance of team formation problems encountered by players in this setting.

Players in these and competitors' games are interested in forming teams where individual desires for roles and team compositions are compatible. A Counter-Strike player who prefers to act as her team's only sniper might prefer a team with no others with the same preference. A League of Legends player who prefers to play a supporting role might want a team with at least one player who needs this support. It would be valuable for player desires to be more consistently fulfilled, since this determines enjoyment of the game and the success of the game studio. I am interested in situations where players have utility over their own role and the composition of roles in their assigned team.

As additional motivation I consider dynamic heterogeneous team formation for robotic urban search and rescue and planetary exploration. Gunn and Anderson observe that "as robots change roles and teams, the overriding goal is to form stable teams" and that "adjusting the roles robots fill on the team can occur when a robot loss or failure is recognized, or when a new robot is encountered and a team merge and redistribution can occur [40]." Now consider a planetary exploration setting, where robots designed by a variety of manufacturers, corporations, private citizens and countries are tasked to work together to accomplish pre-set and not-necessarily-homogeneous goals. Robots far from home must be able to work effectively in a dynamic environment with unpredictable weather and other agents with possibly different goals and motivations [35, 79].

Even when robots share the same utilities for compositions or share the same goal such as mapping unfamiliar terrain [20], developing conditions may cause changes in preferences for certain roles and compositions. A robot low on energy may be qualified to fulfill the role of Driller while working on a short project or when working with other Driller robots, but may reject compositions requiring more resources than it can produce at the time. Similarly, a robot designed to fulfill the Driller role (among others) may have reduced utility for fulfilling this role if its drilling arm is damaged. Additionally, as objectives are completed on the planetary surface, certain compositions may have lowered utility for different agents. Agents with high fuel reserves may have low utility for a composition related to recharging solar fuel cells, for example. Though these robots are designed to work together, they should be able to make decisions which optimize productivity for the individual robots involved.

Modular re-configurable robots for planetary exploration have been considered, with early work by Yim et al. [82], work in cooperative reassembly by Tuci et al. [75], more recent advancements in modular re-configurable robotic systems by Eckenstein and Yim [30] and Ahmadzadeh and Masehian [2], and recent hardware developments such as ModRED, as presented by Baca et al. [11] and Hossain et al. [43]. These advancements allow for individual robots to change and adopt new roles with re-configurations made as needed. Underlying utility for a composition could be used to determine the value of changing to a new role, possibly at the cost of another role.

1.2 What Makes a Good Solution

Though a central authority might wish to maximize total utility for a partition, this goal relies upon the agents' acceptance of the assignment. When agents have autonomy and hedonic utilities, they can and will choose to make local changes for improvement. A partition from the central authority should make such changes unnecessary.

Finding an optimal partition is not a sufficient or worthwhile goal if agents don't stick to the plan. In an MMO, players are unlikely to know or care about the global utility of a partition. They will be more interested in changing the assignment to improve their own utilities. The players are not a captive audience. Should players not find partitions to their liking, they may switch to other games or even *read books* in the worst case [70]. The MMO industry is highly competitive and the players can be quite fickle.

In order to improve acceptance of partitions by the population, finding *stable* partitions needs to be the focus. While I expect agents to make changes which gravitate towards a stable partition, it is worthwhile to partition them such that these changes are easy or unnecessary. If a player is consistently matched to a team which they find unacceptable, the player may quit the game altogether rather than have to improve every assignment offered.

Each of these optimization and stability problems is concerned with a scenario where a central authority would determine both the team assignments of players within the RBHG population and their role assignments within teams. In online games such as League of Legends and DoTA 2, the central authority only controls which teams the players are assigned to. Role assignments are chosen by the players after teams are formed. It has also been observed that, for some stability measures, there may be multiple stable assignments with variable utility levels [69]. A stable assignment is not guaranteed to have an optimal utility even among stable assignments. For this reason, I ultimately focus attention on optimizing the "local world" in which a partition will autonomously form its final matching of roles. I call these optimization goals *optimal expected stability* and *optimal expected utility.* In this dissertation, I consider optimization and stability problems for settings with variable power on the part of the central authority and on the part of the agents.

1.3 Chapter Outline

In Chapter 2, I define the team formation models considered and the optimization and stability goals I seek to achieve in the RBHG setting. I also show that, given a partition of agents to teams, the *optimal composition utility*, *expected stability*, and *expected utility* of matchings for this partition can all be computed in time polynomial in the size of the input.

Chapter 3 outlines related work in optimization and stability for hedonic coalition formation games, the use of roles in object-oriented programming, real-world considerations for roles that players fulfill in online games, and team formation for urban and planetary exploration robots with possibly changing roles.

Chapter 4 provides proofs of computational complexity for several important optimization and stability problems in RBHG. When a central authority has full control over the partition and the matching, I prove that optimization problems concerning finding *perfect*, *max sum* utilitarian, and *max min* egalitarian solutions are all NP-hard in RBHG. For cases in which agents may choose to defect from a given partition and/or matching from the central authority, I prove that finding *Nash stable*, *envy-free and Nash stable*, *individually stable*, *core stable*, and *strict core stable* solutions are all NP-complete problems, while deciding *Pareto optimality* or *contractual strict core stability* of the *grand coalition* is coNP-complete.

Chapter 5 provides two heuristic matchmaking approaches and the results of testing on both synthetic and real world matchmaking data. I introduce *greedy voting* and *greedy location search* heuristic methods for generating RBHG solutions. These methods are compared with respect to *optimal composition utility*, *expected stability*, and *expected utility* against optimized matchings on partitions selected uniformly at random. The methods are validated on real-world data scraped from League of Legends games and on randomly generated RBHG instances with various methods for generating role and composition utilities.

Chapter 6 outlines the results of this work and my plans for future work in recommendation systems for League of Legends and other role based hedonic games.

Chapter 2 Preliminaries

In this chapter, I outline the Roles and Teams Hedonic Game model (RTHG) [69] in Section 2.1 and the generalized Role Based Hedonic Game model (RBHG) [68] in Section 2.2. An instance of RTHG is an instance of RBHG where team size is fixed at an integer m and all possible compositions of that size are considered. The RBHG model allows more flexibility for varying team sizes and allows some compositions to be omitted from consideration. RBHG allows for the consideration of more general settings where certain compositions are not feasible (a sniper team with no snipers) or generally not well accepted by the population (a MOBA team consisting only of support champions). Additionally, I outline the Additively Separable Hedonic Game model (ASHG) [9,16] in Section 2.3. Several stability problems are known to be hard in ASHG. I perform reductions in Section 4.2 to show that these problems remain hard for RBHG.

I consider optimal solutions and stable solutions as goals for matchmaking in these settings. A solution is optimal with respect to some function of agent utilities if no other solution has a better value for that utility function. Examples include maximizing average utility of all teams (MaxSum) or maximizing the utility of the worst-off team in the solution (MaxMin). A solution is stable with respect to some set of possible movements if no agent, or group of agents, can improve utility by defecting from the solution within the confines of these allowable moves. I define several such optimal and stable solution goals in Sections 2.4 and 2.5, and provide reductions to show hardness for these problems in Section 4.1.

2.1 Roles and Teams Hedonic Games

A *Roles and Teams Hedonic Game* (RTHG) instance consists of:

Table 2.1: Example RTHG instance with $|P| = 4, m = 2, |R| = 2$

$\langle r, t\rangle$	$u_{p_0}(r,t)$	$u_{p_1}(r,t)$	$u_{p_2}(r,t)$	$u_{p_3}(r,t)$
$\langle A, AA\rangle$	2	2	0	0
$\langle A, AB\rangle$	0	3	2	2
$\langle B, AB\rangle$	3	0	3	3
$\langle B, BB\rangle$	1	1	1	1

- P: A population of players $\{p_1, p_2, ...p_k\}$ for some integer k.
- m: a team size (I assume that $|P|/m$ is an integer);
- R: A set of roles
- C: A set of compositions, where a composition $c \in C$ is a multiset (bag) of roles from R and $|c| = m$, and where C contains all size m multisets from R. For RTHG instances, it is not necessary to take C as a parameter as it may be generated from m and R.
- $U : P \times R \times C \to \mathbb{Z}$ defines the utility function $u_i(r, c)$ for each player p_i. Unless otherwise specified, I assume that for all $p_i \in P$ and for all $r \in R$, $u_i(r, \{r\}) = 0$, representing an agent's utility for being partitioned to a team by itself. Role and compositions assignments with a positive utility are seen by the agent as an improvement *for that agent* over being partitioned alone, while role and composition assignments with a negative utility are seen as being worse than being left alone.

See Table 2.1 for an example RTHG instance. A solution to an RTHG instance is a partition π of agents into teams of size m and a matching M of agents to roles in R. I denote $\pi(p_i) = t^i$ as the team (set of agents) to which p_i is partitioned and $M(p_i) = r^i$ as the role to which p_i is assigned. The *composition* $c^i \in C$ of t^i is the multiset of roles to which agents in t^i are matched.

2.2 Role Based Hedonic Games

A *Role Based Hedonic Game* (RBHG) instance consists of:

- P: A population of players $\{p_1, p_2, ...p_k\}$ for some integer k.
- R: A set of roles
- C: A set of compositions, where a composition $c \in C$ is a multiset (bag) of roles from R. For some RBHG instances, C may be taken as a parameter. This depends upon the form of the utility function.
- $U : P \times R \times C \to \mathbb{Z}$ defines the utility function $u_i(r, c)$ for each player p_i. Unless otherwise specified, I assume that for all $p_i \in P$ and for all $r \in R$, $u_i(r, \{r\}) = 0$, representing an agent's utility for being partitioned to no team at all.

The *Roles and Teams Hedonic Game* (RTHG) model assumes a fixed team size m which all teams share and that the set of compositions C includes all possible multisets of R [69]. In RBHG, there is no fixed team size and the set C need not include all compositions.

Observe that some compositions may be considered universally unacceptable either by the population (a MOBA team of all healers which has little chance of winning) or a central authority (a military commander forming a sniper team with no snipers). Team size may not have reason to be fixed in real-world scenarios. Even in MOBA games, where team size is usually fixed for each team, some players may join a queue as a preformed "buddy group" needing only a few additional players. A central authority could leverage the RBHG model to find smaller sub-teams to complete these groups.

A solution to an RBHG instance is a partition π of agents into teams and a matching M of agents to roles in R. I denote $\pi(p_i) = t^i$ as the team (set of agents) to which p_i is partitioned and $M(p_i) = r^i$ as the role to which p_i is assigned. The *composition* $c^i \in C$ of t^i is the multiset of roles to which agents in t^i are matched.

Because RBHG is a hedonic game, utility of a player p_i for a partition π and matching M is $u_i(r^i, c^i)$.

2.3 Additively Separable Hedonic Games

An *Additively Separable Hedonic Game* (ASHG) [9,16] instance consists of:

- N: A population of players $\{n_1, n_2, ...n_k\}$ for some integer k.
- V: A vector V of utility functions over all pairs of agents in N, where $v_i(n_j) \in V$ is an integer representing the utility player n_i has for having player n_j on its team. Unless otherwise specified, I assume that $v_i(n_i) = 0$ for all $n_i \in N$, representing an agent's utility for being partitioned to a team by itself.

A solution to an ASHG instance is a partition π of agents into teams. I denote $\pi(n_i) = t^i$ as the team (set of agents) to which n_i is partitioned.

Because ASHG is a hedonic game, utility of a player for a partition π is equal to that player's utility for its team: $\pi(n_i) = t^i$. Let $v_i(t) = \sum_{j \in t} v_i(n_j)$ be the *utility* of player n_i for team t. A player n_i's utility for π is $v_i(t^i)$.

Because some researchers [12,16,36] have dealt with special cases of ASHGs, much of the recent research refers to the definition just given as *general ASHGs*.

When agents in RTHB, RBHG and ASHG have binary preferences for partitions which they either will or will not find acceptable, I call these *accept–reject* cases. In an *accept–reject* instance of each game, an agent will *accept* a partition if it has positive utility for the assignment and *reject* the partition otherwise.

2.4 Optimal Solutions

In this section, I define perfect, MaxSum, and MaxMin solutions for RBHG. I prove that perfect solutions may not always exist.

Perfect solutions for general hedonic games are defined such that each agent is in one of her most preferred coalitions [6]. For RBHG, I define a perfect solution to be one in which each agent gets a most-preferred role and composition pair. Note that, in the general RBHG model, there may be multiple equivalently-valued compositions and roles. Therefore these preferences are not necessarily strict.

Definition 2.4.1. *Given an instance B of RBHG, a* perfect solution (π, M) *is a partition of agents to teams and a matching of agents to roles so that, for each* $p_i \in P$*, where* $M(p_i) = r^i$ *and* $\pi(p_i) = t^i$ *has composition* c^i*,* $u_i(r^i, c^i) = max\{u_i(r', c') : r' \in R \wedge c' \in C\}$*.*

A perfect solution is impossible for some RBHG instances. Consider an RBHG instance where $P = \{\text{Alice}, \text{Bob}\}$. Both Alice and Bob strictly prefer the team composition of ⟨Mage, Assassin⟩ with themselves playing the role Assassin to all other $\langle r, c\rangle$ pairs. No perfect partition is possible.

I consider the following notions of utility optimization.

Definition 2.4.2. *Given an instance B of RBHG, a* MaxSum solution *is one that achieves the maximum value of* $\Sigma_{i<|P|} u_i$*.*

Definition 2.4.3. *Given an instance B of RTHG, a* MaxMin solution *is one that achieves the maximum value of* $\min_{p_i \in P} u_i$*.*

2.5 Stable Solutions

The notion of stability of a partition is one in which no agent has incentive to move from their assigned coalition. Different notions of stability depend on different constraints on agent movements. In this section, I describe the broad notion of stability in hedonic games. I formally define several RBHG stability problems of the form, "Given an instance of RBHG, does a stable solution exists?" I compare these to the related ASHG stability problems and observe known hardness results.

A partition for an RBHG instance is stable when agents have no incentive, or perceived improvement of utility, for changing their assigned role and team. Changing involves some sort of *movement* by the agents.

Any movement begins from a partition π and results in a new partition π'. This change may be made by individual agents or jointly by a group of agents. A partition π of a hedonic coalition formation game is *stable* if, given a set of possible movements, no agents would improve utility by making such changes.

When I say that a player i *moves from* a team t to a team t', I mean that the partition π containing t and t' is modified such that $t := t - \{i\}$ and $t' := t' \cup \{i\}$. This creates a new partition π' in which i is a member of t' and not a member of t. When I say that a team *breaks off* from a partition I mean that these agents *move from* their current teams in the partition and form a new team t' together. When the group of agents breaks off, this creates a new partition π' which includes the team t'.

Individually Rational A partition π is *individually rational* (IR) iff no player can benefit by moving from its team t^i to a team by itself.

In ASHG, π is individually rational iff all players $n_i \in N$ have utility $v_i(t^i) \geq 0$. In RBHG, π is individually rational iff all players $p_i \in P$ have utility $u_i(r^i, c^i) \geq 0$. Generally, we assume that an agent has utility 0 for being on a team by itself.

Nash Stable A partition π is *Nash stable* (NS) iff no player $p_i \in P$ can benefit by moving from its team t^i to another (possibly empty) team t'.

In ASHG, π is Nash stable iff π is individually rational and it holds that for all $n_i \in N$, for all $t' \in \pi$, $v_i(t^i) \geq v_i(t' \cup \{n_i\})$. In RBHG, π is Nash stable iff π is individually rational and it holds that for all $p_i \in P$, for each $t' \in \pi$ having a composition c', for all $r' \in R$, $u_i(r^i, c^i) \geq u_i(r', c' \cup \{r'\})$.

Observation 2.5.1. *[73] Checking whether an* NS *partition exists in an ASHG is NP-hard.*

Individually Stable A partition π is *individually stable* (IS) iff no player $p_i \in P$ can benefit by moving from its team t^i to another (possibly empty) team t' while not making members of t' worse off.

In ASHG, π is individually stable iff π is individually rational and it holds that for all $n_i \in N$, for all $t \in \pi$, if $v_i(t^i) < v_i(t \cup \{n_i\})$ then $v_j(t) > v_j(t \cup \{n_i\})$ for some $j \in t$. In RBHG, π is individually stable iff π is individually rational and it holds that for all $p_i \in P$, for all $t \in \pi$, for all $r' \in R$, if $u_i(r^i, c^i) < u_i(r', c \cup \{r'\})$ then $u_j(r^j, c) > u_j(r^j, c \cup \{r'\})$ for some $j \in t$.

Observation 2.5.2. *[73] Checking whether an* IS *partition exists in an ASHG is NP-hard.*

Core Stable A team t' *blocks* a partition π if each player $i \in t'$ has greater utility for t' than its current team $t^i \in \pi$. A partition π which admits no blocking coalition is said to be *in the core* or *core stable* (CS). If the core is empty, this means that there are no core stable partitions.

In ASHG, a team t' blocks a partition π iff there is a set $N' \subseteq N$ where t' is a team consisting of all agents in N' and $v_i(t') > v_i(t^i)$ for all $n_i \in N'$. In RBHG, a team t' having a composition c' blocks a partition π iff there is a set $P' \subseteq P$ and an assignment of agents in P' to the bag of roles c such that $u_i(r^i, c) > u_i(r^i, c^i)$ for all $p_i \in P'$.

Observation 2.5.3. *[9, 73] Checking whether a non-empty* CS *partition exists in an ASHG is NP-hard.*

Strict Core Stable A team t' *weakly blocks* a partition π if each player $i \in t'$ has greater or equal utility for t' compared to its current team $t^i \in \pi$ and at least one player $j \in t'$ has greater utility for t' than its current team $t^j \in \pi$. A partition π which admits no weakly blocking coalition is said to be *in the strict core* or *strict core stable* (SCS). If the strict core is empty, this means that there are no strict core stable partitions.

In ASHG, a team t' weakly blocks a partition π iff there is a set $N' \subseteq N$ where t' is a team consisting of all agents in N', $v_i(t') \geq v_i(t^i)$ for all $n_i \in N'$, and $v_j(t') > v_j(t^j)$ for at least one $n_j \in N'$. In RBHG, a team t' having a composition c' weakly blocks a partition π iff there is a set $P' \subseteq P$ and an assignment of agents in P' to the bag of roles c such that $u_i(r^i, c) \geq u_i(r^i, c^i)$ for all $p_i \in P'$ and $u_j(r^j, c) > u_j(r^j, c^j)$ for at least one $p_j \in P'$

Observation 2.5.4. *[9, 73] Checking whether a non-empty strictly core stable partition exists in an ASHG is NP-hard.*

Contractual Strict Core Stable A partition π is said to be *in the contractual strict core* or *contractual strict core stable* (CSCS) iff any *weakly blocking* team t' makes at least one player $n_j \in N \backslash t'$ worse off when breaking off.

In ASHG, a player $n_j \in N$ is worse off when a *weakly blocking* team t' breaks off if some agent $n_i \in t'$ was formerly on $t^j \in \pi$ and $v_j(t^j - \{n_i\}) < v_j(t^j)$. In RBHG, a player $n_j \in N$ is worse off when a *weakly blocking* team t' breaks off if some agent $p_i \in t'$ was formerly on $t^j \in \pi$ in a role r^i and $u_j(r^j, c^j - \{r^i\}) < u_j(r^j, c^j)$.

Observation 2.5.5. *[9] Verifying whether the partition is* contractual strict core stable *in ASHGs is coNP-complete.*

Pareto Optimal A partition π is Pareto optimal (PO) iff there is no partition π' such that each agent has utility greater than or equal to their utility for π and at least one agent has greater utility for π' than for π [9].

In ASHG, a partition π is Pareto optimal iff there is no partition π' such that $v_i(t'^i) \geq v_i(t^i)$ for all $n_i \in N$ and $v_j(t'^j) > v_j(t^j)$ for at least one $n_j \in N$. In RBHG or RTHG, a partition π is Pareto optimal iff there is no partition π' such that $u_i(r'^i, c'^i) \geq u_i(r^i, c^i)$ for all $p_i \in P$ and $u_j(r'^j, c'^j) > u_j(r^j, c^j)$ for at least one $p_j \in P$.

Observation 2.5.6. *[8, 9] Verifying whether a given partition π is* Pareto optimal *in ASHGs is coNP-complete.*

Envy Free A partition π is *envy free* (EF) iff no player has utility for her team that is less than her utility for another agent's team.

In ASHG, π is EF iff no player $n_i \in N$ has utility $v_i(t^i) < \Sigma_{k \in t^j} v_i(k)$, for some player $n_j \in N$ on team $t^j \in \pi$. In RBHG, π is EF iff no player $p_i \in P$ has utility $u_i(r^i, c^i) < u_i(r^j, c^j)$, for some player $p_j \in P$ on team $t^j \in \pi$ in role r^j.

Observation 2.5.7. *[8] Checking whether there exists a partition which is both envy free and Nash stable in ASHGs is NP-complete even if preferences are symmetric.*

2.6 Expected Utility and Stability

Given a partition of agents to a particular team t, the utilities of the agents within t over the roles within some composition c can be stored in a $|t|$ x $|t|$ matrix. One such matrix for each of C compositions represents the utilities of the agents of t for all compositions.

In this section, I define several optimization objectives given a team t. An *optimal utility* matching for a team t given a composition c is one for which total utility is maximized for that team and composition. I define an *optimal composition* for a team t as a composition for which *optimal utility* is maximized. For *accept–reject* instances of RBHG, I define a *stable composition* as one for which there is a matching of agents to roles which each agent on the team *accepts* (has a positive utility for).

I define *candidate compositions* which may result from a team being matched to roles given the utilities of its members for roles and compositions. I define *reachable compositions* as compositions which can be formed by a team given their preferences for roles. I define *advocated compositions* as those compositions for which at least one agent on the team has a positive utility.

I define the *expected utility* of a team t as the average *optimal utility* across a set of *candidate compositions*. I additionally define *expected stability* for *accept–reject* instances of RBHG as percentage of *candidate compositions* which are also *stable compositions*. Finally, I define *optimal expected utility* and *optimal expected stability* as those partitions for which *expected utility* and *expected stability* are respectively maximized.

Definition 2.6.1. *Given a partition π of an instance B of RBHG, a team $t \in \pi$, and a composition $c \in C$, the* optimal utility *of t given c is the maximum sum of the utilities of all agents, maximized over all matchings of agents to roles in c.*

For accept–reject *instances of RBHG, a* stable composition *of t is a composition ${c_t}^o \in C$ for which an* optimal utility *assignment of t has $u_i(r, c) > 0$ for all $p_i \in t$.*

An optimal composition *of t is a composition ${c_t}^o \in C$ for which the* optimal utility *of t for all $c \in C$ is maximized.*

Observation 2.6.2. *The* optimal utility *of a team t given a composition c can be computed in time $\mathcal{O}(|t|^3)$ by application of the Kuhn Munkres algorithm for optimizing square matrices [31]. Similarly, it can be determined whether or not a given composition c is a* stable composition *for a team t.*

The optimal composition *${c_t}^o \in C$ of t can be identified in time $\mathcal{O}(|C| \cdot |t|^3)$ by application of the Kuhn Munkres algorithm once for each $c \in C$.*

Without a central authority to assign agents to roles within a team t, agents in t will either jointly accept a composition c or one or more agents will defect from the team t. Therefore, I consider evaluating the quality of a team t by either its *expected utility* or *expected stability* across the set of *candidate compositions*.

Definition 2.6.3. *A* candidate composition *c for a team t is a multiset of roles in R to which agents in t may be matched.*

A reachable composition *is a* candidate composition *c for which there is an onto mapping f from the the players in t to the multiset of roles c so that, for each player p_i, there is some $c \in C$ such that $u_i(f(p_i), c) > 0$. Observe that any* stable composition *is also a* reachable composition *but that the converse is not necessarily the case.*

An advocated composition *c for a team t is a* candidate composition *for which $u_i(r, c) > 0$ for at least one agent in $p_i \in t$, for at least one role $r \in c$. Observe that any* stable composition *is also an* advocated composition *but that the reverse is not necessarily the case.*

The number of *reachable compositions* is of interest in cases where agents may be assigned to a matching which they do not necessarily accept. In the League of Legends setting, this could occur if the match making timer runs out while players are still making adjustments. In the case of robots on Mars, the time limit may be due to depleting battery power requiring agents to act before an acceptable matching is found. Notably, a composition may be reachable by a team of agents *even if no agent accepts the composition.* Yet, because these forced moves can occur, it is important to consider optimization over this set of possible results.

An advocated composition can be seen as one which at least one agent would *advocate* for and therefore remain in when reached. This can occur in the case of League of Legends when a player declares "I like this, let's play!" or otherwise stops changing roles. The number of *advocated compositions* is valuable to consider when agents have fewer time limitations and are unlikely to accept a composition prematurely. While it is possible for the League of Legends timer to run out, players are capable of quitting a team and rejoining the queue before this occurs if the composition is particularly bad. Similarly, a group of Mars robots may choose to defect from a team before batteries start to run out if early bargaining rounds are failing. The set of *reachable* and *advocated compositions* consists of those which are not only reachable by the team but would also be advocated by at least one member of the team.

Definition 2.6.4. *Let qs_t be the number of stable compositions for t and qa_t be the number of candidate compositions for t. The* expected stability *of t is the ratio qs_t/qa_t for $qa_t > 0$ or 0 if $qa_t = 0$.*

Let ua_t be the sum of optimal utilities for all qa_t candidate compositions for t. The expected utility *of t is the ratio of ua_t/qa_t for $qa_t > 0$ or 0 if $qa_t = 0$.*

I define the following optimization problems.

Definition 2.6.5. *An* optimal expected stability partition *π of and instance B of RBHG is one in which the average expected stability for all $t \in \pi$ is maximized.*

An optimal expected utility partition *π of an instance B of RBHG is one in which the average expected utility for all $t \in \pi$ is maximized.*

In this dissertation, I evaluate *expected utility* and *expected stability* both over the set of *reachable compositions* and the set of *reachable* and *advocated compositions.* Experimental procedures and results are detailed further in Chapter 5.

Chapter 3 Related Work

A coalition formation game consists of a set of agents, a preference profile of the agents, and the set of possible partitions, where a partition is a set of teams (subsets of agents). In economics, Drèze and Greenberg introduced coalition formation games to model situations where agents join teams to collaboratively produce goods for themselves [29]. An agent's goal in this setting is to optimize consumption of goods within the partition.

In a *hedonic* setting, agents are interested in optimizing consumption for their own teams and not others. The game is hedonic in that an agent's preference for a partition is determined only by the coalition to which the agent is assigned. Banerjee et al. [13] and Bogomolnaia and Jackson [16] initiated much of the work in the hedonic coalition formation game setting, which now spans a variety of models. A recent survey by Peters and Elkind follows much of this literature [56], albeit not addressing the recent addition to the literature of RBHGs. Hedonic coalition formation game models have been suggested for a variety of multi-agent settings, including distributed task allocation in wireless agents [59], communications networks [61], vehicular networks [62,81], adaptive smart grid management [49], and federation formation among cloud providers [39], among others.

For general hedonic games, Ballester showed that determining whether there is a non empty core is NP-complete [12]. Sung and Dimitrov considered farsighted stability in hedonic games, where agents avoid defecting from a team if the defection would induce a worse change by agents on their new or former team [27]. In other words, agents in this setting will settle upon a less-desired partition so as to avoid an even worse partition caused by "rocking the boat" with alterations. Sung and Dimitrov showed that core stable and Nash stable solutions are farsighted stable while individually stable and contractually individually stable solutions are not necessarily [27].

For hedonic games in which the set of teams is individually rational, Ballester found that checking the existence of Nash stable and individually stable partitions and checking the existence of a non empty core are all NP-complete [12]. Sung and Dimitrov showed that these problems, in addition to the problem of determining whether the strict core is not empty, remain NP-complete for hedonic games with additively separable preferences [73]. Additively separable hedonic games (ASHG) [16] allow for agents to place values on each other, making the agent population heterogeneous. The value an agent places on its coalition in such a game is the sum total value it gives other agents in its coalition. Aziz et al. showed that finding envy-free and Nash stable partitions, max sum partitions, and max min partitions are all NP-hard [8], while finding checking if the strict core is non empty and checking whether the grand coalition is Pareto optimal are both coNP-complete [9]. Aziz et al. provided polynomial time algorithms for finding contractually individually stable solutions [9] and for finding partitions satisfying envy-freeness and individual rationality [10].

The ASHG model considers agent-to-agent valuation, but these values are fixed for any given agent-to-agent relation. ASHGs do not consider the context of the composition an agent is in. In RBHG, values are placed on team compositions and roles rather than individual agents. An agent may be highly desirable when paired with other suitable agents, but undesirable without the proper composition. Each agent has a variable role in RBHG and has preferences over which role to select for itself given a team composition.

In the *friends and enemies* variants of additively separable hedonic games, each agent considers each other agent as either a friend or an enemy [28, 72] rather than having a more complex utility function. Dimitrov et al. showed that a strict core stable solution can be found in polynomial time when agents value partitions based on an *appreciation of friends* with no regard to enemies, and that a core stable solution can be found when utility is based on *aversion to enemies*. Lang et al. introduced the notion of neutral relationships to this problem [50], showing that determining whether a Nash stable partition exists is NP-complete even when agents only characterize others as friends, enemies, or neutral parties.

Aziz, Brandt and Harrenstein introduced *fractional hedonic games* [7], where utility is viewed as an agent's average utility for all agents on its team. In this setting, Aziz et al. showed that computing a core stable partition for an instance with symmetric preferences is NP-hard, while checking if a given partition is core stable is coNP-complete. However, for certain special cases of fractional hedonic games, the core is guaranteed to be non-empty. Recent work by Bilò et al. considers the *price of stability* in fractional hedonic games [14]. In general, the price of stability for a hedonic game instance is a lower bound on social welfare achieved by some global utility function (such as *max sum* or *max min* utility) given the best case (in terms of that utility function) of some stability measure (such as *Nash stability* or *individual stability*). The similar *price of anarchy* is a lower bound on the utility function given any solution satisfying the stability measure [53].

Brânzei and Larson introduced *social distance games* in which utility is measured by an agent's *social distance* to other agents on its team [18]. Social distance between two agents is defined in economics as the distance between two agents by various social measures (geographical, cultural, interpersonal similarity, etc.) [3, 52]. What is important for the *social distance games* model is that there is some measure by which to gauge how *close* two agents are to one another in terms of edge distance in a unweighted graph containing no loops. An agents' utility is hedonic in this setting, defined in terms of its closeness to the other agents on its team. In this regard the model is similar to *fractional hedonic games* in that the average utility for other members of the team is taken to form the total utility [7]. While agents in *role* *based hedonic games* are assumed to be anonymous, social distance games assume that connections between agents are defined.

When matching hedonic agents into pairs rather than larger teams, the problem of finding a stable assignment reduces to the stable roommates problem [47, 48], for which fast algorithms exist when a stable matching does exist. It has been observed that the likelihood of there being a stable roommate assignment diminishes as the population size increases [57]. More recently the problem of finding near stable roommate assignments has been considered [1, 15]. This relates closely to my goal of optimizing *expected stability* of RBHG instances. Expected stability is a measurement of the likelihood that a team of players will stabilize on a matching of roles. In a multiplayer online battle arena such as League of Legends, players match themselves into roles after they are partitioned into teams by the central authority. As a result, there is a lack of control over precisely which roles the agents will adopt in the final solution. The goal of optimizing expected stability is to increase the ratio of stable compositions to the set of compositions a team may consider in this second phase.

Anonymous hedonic games have been considered where agents have preferences over the size of the team but not over whom they are matched with [12,13]. The *group activity selection problem* introduced by Darmann et al. is a variant of anonymous hedonic games where agent preferences are dependent upon both the size of the team and the activity being performed by the team [25].

In *role based hedonic games* (RBHG), agents are heterogeneous while the group activity is fixed for a given instance. An RBHG agent holds preferences over its own role and the roles of its teammates. Preferences are not necessarily based upon the size of a team. In these ways, RBHG considers the problem of anonymous agents in a different way than existing work that focuses on differences in team sizes and group activities.

Work in hedonic coalition formation for large networks has considered a different approach to handling anonymity. Taking, for example, the case of researchers forming research groups: Hoefer et al. have observed that agents may choose not to form groups with agents they do not already know [41]. Hoefer et al.'s work considers an agent to have visibility only for its existing base of known agents and the resulting issues in locality. Here, agents can learn new contacts over the course of forming teams, maintain a permanent contact list, and reference temporary contacts with teams which include members who are not yet a part of their permanent contact list. In this way, the partially anonymous game may slowly transform into a normal hedonic coalition formation game without anonymity, where all agents have finally added one another to their contact lists. This is a different problem where anonymity is considered to be a problem to overcome.

In RBHG, anonymity is a fact of the game. Agents do not express their utilities in the RBHG setting in the context of *who* is on the team, and thus do not build up a local set of contacts. It is reasonable to combine the two problems, where agents have a set of known agents and otherwise have utilities for roles and compositions in absence of local contacts. This takes the two "pure" scenarios and creates a more expressive system to handle both preferences for locally known agents and preferences for "types of colleagues." In the research team formation problem this would be especially helpful for young researchers who have not yet built a full portfolio of contacts. Alternatively, mock team formation using RBHG utilities could be used to make recommendations for new local contacts, allowing for more direct evaluation of a smaller subset of the population.

Sometimes a model of coalition formation can be seen as an overlapping of existing models. Consider the *anonymous stable invitation problem* [51] introduced by Lee and Shoham. This setting combines several variants of hedonic coalition formation games into a single problem, where an organizer for an event wishes to invite the maximum size subset of agents who will all attend the event together. The problem could be extended to maximizing the total value (perhaps in terms of donations) of the individuals invited, though it can be assumed that all agents have equal value. Again, this setting may have utility for both the number of individuals invited and for the specific individuals who will be in attendance. Rather than attempting to form a large number of teams, the goal is to determine the largest stable coalition (stable in that all agents in the coalition will accept it). The authors observe that this new problem bears similarities to both *additively separable hedonic games* and *anonymous hedonic games*. Combining the two problems, this work allows for the possibility of utilities over both individual agents and the size of the team. The goal of the game is analogous to determining the maximum sized *individually rational* team which no agent would reject in favor of sitting out of the event. If the problem were extended to inviting agents to several different events, it could be viewed as an overlay of *additively separable hedonic games* and the *group activity selection problem*.

Recent work investigates problems in hedonic games caused by the presence of a moral hazard: an economic phenomenon where one agents takes on production responsibilities for which another agent bears the risk [76]. When teams can only make balanced transfers of production after initial assignment, stability is negatively affected. When teams cannot make production transfers at all, stability is positively affected but efficiency (utility) is negatively effected. This is an interesting setting to consider for RBHG problems, and relates to the problem of ensuring honest reporting of preferences and requirements (such as voter manipulation in elections [54, 84] or dishonest budgets reporting [23]).

Recent work related to hedonic games considers the mechanism design problem [80] for additively separable hedonic games where all utilities are non-negative. In this problem, the desirable properties considered for a matchmaking mechanism are to ensure that the mechanism creates no incentive for an agent to misreport its preferences and that the mechanism ensures optimal social welfare and fairness. I observe that problems in this setting have a trivially Pareto optimal solution consisting of the *grand coalition*, where all agents are assigned to the same team. This is due to the assumption that all utilities are non-negative and additively separable. However, the more interesting cases for my work are when team size has a maximum value less than the number of players.

The role of an agent in the RBHG model can be seen either as the task the agent is performing, the way the agent performs a task, or some other conceptual quality such as the color of the agent's shoes. What is important is that an agent may fulfill different roles and may have utility for different roles depending upon the composition of roles it is presented with in a team. The notion of roles has been applied previously to object-oriented database programming. Albano et al. developed *Fibonacci* as an object-oriented database programming language in which data objects are manipulated through the changing roles that they fill [4]. Gottlob et al. extended this work to model additional features, such as having *qualified roles* which are multiple distinct instances of the same role (a Project Manager role for each of multiple projects, for example) [38]. Gottlob's use of the term *role* differs from the RBHG model in that I consider an agent to, at any one time, fulfill a single role rather than multiple roles at once. It is similar in that both works consider the possibility that a single entity may change roles over time, fulfilling different positions while still retaining basic qualities (in the RBHG case, utilities for different roles and compositions). When an agent might fulfill multiple roles at once, or provide a set of skills to complete different tasks, Tran-Tranh et al. introduced the *coalition skill vector* model for coalition formation games where each agent possesses a set of skills, there is a set of tasks to perform, and the value of a partition is based upon the number or total payoff of tasks the teams can jointly complete [74]. In the RBHG model, I am concerned with the single role that an agent is fulfilling at any given time and possible changes from one role to another when determining the single composition of the team.

Related work in the social aspect of roles in online games considers the phenomenon of *gender swapping*, where players choose avatars of a gender (here seen as male or female) other than one that they are commonly associated with (either personally or by society as a whole) [67]. This work investigates both the reasons that players may engage in gender swapping and what can be predicted about their in-game spending habits. Additional work considers the impact of *prosocial behavior* in online games [78], both in general and as it relates to gender swapping. Prosocial behavior includes behaviors such as sharing, volunteering time and money to others, and generally taking actions in an effort to help others or society as a whole [5,19,55]. This work relates to determining predictable utility functions which agents may hold for roles and compositions in RBHG settings. For example, an agent who exhibits prosocial behavior may be more inclined to accept a variety of role assignments if it will help a team achieve a preferred composition. Another agent may select avatars based upon aesthetics rather than just the roles that they fulfill. Consider a case where one subset of agents selects League of Legends champions based upon the set of tactical roles they fulfill (Support, Attacker, Defender, etc.) and their related tactical compositions, another subset considers the colors of their uniforms (Teal, Mauve, Salmon, etc.) and their related aesthetic compositions, and still another considers both aspects when evaluating a team. Addressing these different types of utilities in practice may require multi-objective utility functions which consider roles and compositions of various definitions as well as other aspects of avatars being selected. In this dissertation, I consider the simpler problem of having utilities only for role and composition pairs from a single set of roles that all agents agree upon.

Consider the following setting. In capstone computer science courses, students are sometimes grouped into equal size project teams. For a team of five students, one student may prefer a team of two skilled programmers, one designer, and two writers. Her second choice might be one programmer, two designers, and two writers.

In the first case, the student wants to be one of the two programmers and *not* the only designer. In the second, she wants to one of the two designers, and definitely *not* the only programmer.

This problem can be modeled as an RBHG. The ASHG model allows students to express utility values for each other, but ASHG preferences are *context-free* agent-to-agent assessments. Alice may wish to join Bob's coalition when he needs a programmer, but not when he needs a writer. In RBHG, an agent need only express preferences on which roles and compositions she prefers. This evaluation may be easier to accurately poll.

In online games such as League of Legends and Defense of the Ancients 2 [58], where players select avatars having different skills and abilities, success of a team can depend on which roles are filled by the subset of avatars selected. While players are largely anonymous, preferences in terms of roles performed by teammates to achieve some objective can be described. This adds to what it means to be an agent. Rather than having agents who are immutable objects, a single agent can be perceived by other agents to be "worth" different amounts to another agent on her team depending upon what role she is fulfilling. This task-dependent agent valuation also has application with distributed task allocation in wireless agents [60].

For instances where $|C| \cdot m$ is smaller than $|P|$, where m is the maximum size of a composition in $|C|$, the required input data for RBHG instances will be smaller than the required input for ASHG. Input for an ASHG instance requires each agent to hold a specific utility for each other agent within the population. This could be represented as a $|P| \times |P|$ matrix of utility values, U, where $U[i, j]$ is the utility that p_i holds for p_j. In RBHG, the input can be represented as a $(|C| \cdot m) \times |P|$ matrix. While there are millions of players in *League of Legends* [46], there are only around 5 basic roles to potentially fill (AD Carry, Support, Mage, etc.) and a maximum team size of 5. The input required for team formation in this setting will be orders of magnitude smaller in RBHG than if this game were treated as an ASHG. In the case where utilities are additively separable, I show in Chapter 4 that an instance of ASHG can be represented as an instance of RBHG. This use of an additively separable utility function over the available roles can reduce the necessary size of an RBHG instance even more, when applicable.

Stability is a property which, in certain settings, we expect to be generated naturally by the agents. That is, should the agents be allowed to either accept or reject a partition, they will only choose to accept a partition which is stable. This relates to a game such as *League of Legends* and *Defense of the Ancients 2* where players have agency and may choose not to play if a team composition is undesirable (recall that *books* are an option). In these games agents are only partitioned to teams by the central authority and are not automatically matched to roles. Therefore, even if agents are assigned to a partition which contains an optimal *and* stable matching of roles within the teams, it is not guaranteed that the agents will stabilize on that particular matching. This observation directs the goals of optimizing *expected stability* and *expected utility* of a partition.

Consider a team of Mars robots weighing the pros and cons of removing Johnnybot's drill arm and replacing it with a high-definition camera. Perhaps the reconfiguration will help to achieve a single composition currently desired, but would significantly reduce the ability of the team to complete other objectives also considered valuable. By optimizing the expected utility or stability of a team in terms of the abilities of robots to fulfill certain roles and their independent valuations of certain compositions (roles needed to perform certain important tasks), a team will be more likely to remain stable against shifting needs and opportunities. When considering a single team comparing two possible utility functions (one with Johnnybot's drill arm, and one with the camera), I can compute the *optimal composition utility*, the *expected utility* and the *expected stability* for both configurations in time polynomial in the size of the input. This would allow for a fast comparison between working with the existing configurations and the expected value of making the proposed change. In this dissertation, I consider not only the likelihood that a team of agents will stabilize on a single productive composition but that over time the agents will work well together in a variety of circumstances. While I do not consider the ethical implications of removing Johnnybot's arm, quickly determining which modifications to make could improve productivity of robots on a distant world and reduce conflict over the limited resources.

Chapter 4 Complexity Results

In this chapter, we prove that determining whether or not there exists a perfect solution of an RBHG instance is NP-complete and that the decision problems related to finding MaxSum and MaxMin solutions are both NP-complete. We prove that verification of *Pareto optimality* and *contractual strict core stability* of the *grand coalition* are both coNP-complete problems in RBHG, and that the remaining stability problems considered in this dissertation are all NP-complete for RBHG.

4.1 Complexity of Optimization

Definition 4.1.1. *An instance of* Special RBHG *is an instance of RBHG such that for each agent* $p_i \in P$*, each* $c_j \in C$*, and each* $r_k \in c$*,* $u_i(r_k, c_j) \in \{0, 1\}$ *and* $u_i(r_k, c_j) = 1$ *only if* c_j *is* uniform, *namely it consists of* $|c|$ *copies of a single role* r*.*

In other words, each agent finds some non-empty set of single-role team compositions acceptable (utility 1), and no other types of team compositions acceptable.

Definition 4.1.2. *The language* Perfect RBHG *consists of those instances of RBHG for which a perfect solution exists, and* Perfect Special RBHG *consists of those instances of* Special RBHG *for which a perfect solution exists.*

In *Special RBHG* instances, the question of a perfect solution reduces to the problem of finding a MaxMin solution, or the decision problem of whether there's a partition with MaxMin value m.

Consider the Exact 3Cover problem:

GIVEN a set $S \subseteq \mathcal{P}(\{1, ..., q\})$ where all elements of S have size 3,

IS THERE a subset $T \subseteq S$ such that T partitions $\{1, ..., q\}$?

For example, given $q = 6$ and $S_1 = \{\{1, 2, 4\}, \{3, 5, 6\}, \{3, 4, 5\}\}$, there is an exact three cover $S_1' = \{\{1, 2, 4\}, \{3, 5, 6\}\}$. Given $q = 6$ and $S_2 = \{\{1, 2, 4\}, \{4, 5, 6\}, \{3, 4, 5\}\}$, no exact three cover exists. Note that though all elements in $\{1, ..., q\}$ can be found in elements of S_2, an exact cover requires that each element be found exactly once.

EXACT 3COVER is NP-complete [37].

Theorem 4.1.3. PERFECT SPECIAL RBHG *is NP-complete.*

Proof. To show that PERFECT SPECIAL RBHG is in NP, consider the following NP algorithm. Given an instance of PERFECT SPECIAL RBHG, guess a partition and evaluate its MaxMin value. To compute the MaxMin value, compute the utility of each of the at most $|P|$ teams (time $\mathcal{O}(|P| \cdot l)$ for each coalition, where l is the complexity of table lookup for an individual's utility for a particular team and role), stopping and rejecting if any coalition has utility 0, else accepting.

This checking is in time polynomial in the size of the input.

To show NP-hardness, we show that EXACT 3COVER $\leq_m^P$ SPECIAL PERFECT RBHG. In other words, given an instance $E = \langle q, S \rangle$ of EXACT 3COVER, we construct an instance R_E of SPECIAL PERFECT RBHG such that $E \in$ EXACT 3COVER iff $R_E \in$ SPECIAL PERFECT RBHG.

R_E will have the property that, for each agent $p_i \in P$, the only acceptable teams are uniform, i.e., consist of $|t|$ copies of a single role. Thus, the question is whether they can be assigned to an acceptable team; the role for that team will be acceptable.

Consider $E = \langle q, S \rangle$. For each set $s_i \in S$, R_E will have a role $r_i \in R$ and a corresponding team composition $c_i = \{r_i, r_i, r_i\}$. $P = \{1, ..., q\}$. The desired team size is $m = 3$. Each agent p_j desires those team compositions s such that $j \in s$.

There is an exact three cover of $\{1, ..., q\}$ iff there is an assignment of agents to teams of size three such that each team corresponds to an element of S.

Consider an exact three cover $X \subset S$. Then for each $j \leq q$ there is an $s_i \in X$ such that $j \in s$. The exact three cover X corresponds to a partition π for R_E that assigns an acceptable team to each agent. The matching M is implicit with each agent assigned to a unique acceptable role within its team $t \in \pi$. There is only one acceptable composition for that set of agents, namely $c_i = \{r_i, r_i, r_i\}$.

Now assume that there is a perfect solution (π, M) for R_E. Then for each $i \leq |P|$ there is a $t^i \in \pi$ having a composition c_i and such that $u_i(r_i, c_i) = 1$. The partition π and matching M corresponds to a solution $X \subset S$ that is an exact three cover of S.

Therefore, the PERFECT SPECIAL RBHG problem is NP-hard. ❑

Definition 4.1.4. *The language* MAXSUM RBHG *consists of pairs $\langle B, k\rangle$, where B is an instance of RBHG, k is an integer, and the MaxSum value of B is $\geq k$. We denote the MaxSum value of a solution (π, M) as MaxSum(π, M).* MAXSUM SPECIAL RBHG *consists of those instances of* Special RBHG *for which the MaxSum value is $|P|$.*

Definition 4.1.5. *The language* MAXMIN RBHG *consists of pairs $\langle B, k\rangle$, where B is an instance of RBHG, k is an integer, and the MaxMin value is $\geq k$. We denote the MaxMin value of a solution (π, M) as MaxMin(π, M).* MAXMIN SPECIAL RBHG *consists of those instances of* Special RBHG *for which the MaxMin value is $|t|$.*

Theorem 4.1.6. MAXMIN SPECIAL RBHG *and* MAXSUM SPECIAL RBHG *are both NP-hard.*

Proof. A *Special RBHG* solution (π, M) for B is perfect iff $\sum_{p_i \in P} u_i(r^i, c^i) = |P|$ iff MaxMin$(\pi, M) = |t_{min}|$, where t_{min} is the team having minimum utility, iff $\langle B, |P|\rangle \in$ MAXSUM RBHG iff MaxSum$(\pi, M) = |P|$ iff $\langle B, |t_{min}|\rangle \in$ MAXMIN RBHG. Therefore MAXMIN SPECIAL RBHG and MAXSUM SPECIAL RBHG are both NP-hard.

❑

Corollary 4.1.7. *The general cases of* PERFECT RBHG, MAXSUM RBHG, *and* MAXMIN RBHG *are all NP-hard.*

Proof. Observe that the reductions given for Perfect Special RBHG, MaxSum Special RBHG, and MaxMin Special RBHG are also reductions for Perfect RBHG, MaxSum RBHG, and MaxMin RBHG.

Therefore the general cases of Perfect RBHG, MaxSum RBHG and MaxMin RBHG are all NP-hard. ❑

4.2 Complexity of Stability

Definition 4.2.1. *The language* NS RBHG *consists of instances of RBHG for which there exists a Nash stable solution.*

Theorem 4.2.2. NS RBHG *is NP-complete.*

Proof. To see that NS RBHG is in NP, observe that Nash stability for a single RBHG agent can be verified in time $O(|P| \cdot |R| \cdot l)$, where l is the complexity of table lookup for an individual's utility for a particular team and role. Verifying the property for all agents can be done in time $O(|P|^2 \cdot |R| \cdot l)$. To verify Nash stability for a single agent, check if the agent can improve utility by moving from its current team to another team (with any of the possible roles) or by changing from its current role to another role on the same team. In the worst case, where every agent is on a team by itself, this requires consideration of each of the $|P|$ possible teams and each of the $|R|$ possible roles on each team. Verifying the property for all agents requires performing the above test for each of the $|P|$ agents.

Next, we construct a reduction, f, from NS ASHG to NS RBHG. Let A be an instance of NS ASHG with a population N and utility function vector V.

We define $f(A) = (P, R, C, U)$ to be an instance of RBHG. We set $P = N$ and $R = \{r_1, ..., r_{|N|}\}$. The set C is not constructed explicitly in this case, nor is utility for each $\{r, c\}$ pair explicitly stored. Instead, utility of an agent for a team is defined by an additively separable function over the roles in the composition to which it is assigned.

An agent assigned to its own role in a team by itself has utility $u_i(r_i, \{r_i\}) = 0$. Define MaxAbsValue(A) as the maximum absolute value of utility for any agent $p \in P$. An agent assigned to a role $k \neq i$ has utility $u_i(r_k, c) = -\text{MaxAbsValue}(A) \cdot |P| - 1$, for any composition c. For teams of size $m \geq 2$ set $u_i(r^i, c) = \Sigma_{j \in c} v_i(j) = \Sigma_{j \in c} u_i(r^i, \{r^i, r^j\})$.

Observation 4.2.3. *The only partitions with positive values consist of coalitions where, for each $p_i \in P$, p_i is assigned to role r^i.*

Let $f(A)$ be in NS RBHG and let (π, M) be a Nash stable solution of $f(A)$.

From (π, M) we construct a partition π' of A using f^{-1}. Since (π, M) maps each $p_i \in P$ to the role r^i representing $f^{-1}(p_i) = n_i \in N$, we have that $f^{-1}(\pi, M) = \pi'$ is a well-defined partition of A.

To show that π' is a Nash stable partition of A, consider the following proof by contradiction. Suppose there were an agent $n_i \in N$ and a team $t' \in \pi'$ such that $v_i(t' \cup \{n_i\}) > v_i(t^i)$. Consider that the corresponding agent $p_i \in P$ in a team $t \in \pi$ with composition c will have $u_i(r^i, c \cup \{r^i\}) > u_i(r^i, c^i)$. This contradicts the premise that (π, M) is a Nash stable solution of $f(A)$. Therefore if $f(A)$ is in NS RBHG then A is in NS ASHG.

Now let π be a Nash stable partition of A. Let $(\pi', M) = f(\pi)$ be the corresponding solution in $f(A)$. For each agent $p_i \in P$, $u_i(r^i, c^i) = v_i(t^i)$ where $t^i \in \pi$ is composed of the agents represented by the roles in c^i.

By the same argument as in the previous case, we get that (π', M) is also Nash stable. Therefore if A is in NS ASHG then $f(A)$ is in NS RBHG.

Therefore $f(A)$ is in NS RBHG iff A is in NS ASHG. Thus, we have shown that f is a reduction from NS ASHG to NS RBHG.

❑

Definition 4.2.4. *The language* EF NS RBHG *consists of those instances of RBHG for which there exists an envy free Nash stable solution.*

Theorem 4.2.5. EF NS RBHG *is NP-complete.*

Proof. To see that EF NS RBHG is in NP, we observe that envy-freeness and Nash stability for a single RBHG agent can be verified in time $O(|P|^2 \cdot |R|)$. Verifying the property for all agents requires time $O(|P|^3 \cdot |R|)$. To verify Nash stability for a single agent, check if the agent can improve utility by moving from its current team to another team (with any of the possible roles) or by changing from its current role to another role on the same team. In the worst case, where every agent is on a team by itself, this requires consideration of each of the $|P|$ possible teams and each of the $|R|$ possible roles on each team. To verify envy-freeness requires an additional $|P|$ comparisons, where the agent checks if it would improve utility for being in the same role and composition as each of the other agents. That is, rather than joining each other team and seeing if utility improves, we swap roles and team positions between the agent and each of $|P| - 1$ other agents to see if utility improves with any such swap. Verifying the two properties for all agents requires performing the above test for each of the $|P|$ agents.

Next, we construct a reduction, f, from EF NS ASHG to EF NS RBHG. Let A be an instance of EF NS ASHG with a population N and utility function vector V. We use the same function f as in Theorem 4.2.2 to generate an instance $f(A) = (P, R, C, U)$ of RBHG.

Let $f(A)$ be in EF NS RBHG and let (π, M) be an envy free Nash stable solution of $f(A)$.

From (π, M) we construct a partition π' of A using f^{-1}. Since (π, M) maps each $p_i \in P$ to the role r^i representing $f^{-1}(p_i) = n_i \in N$, we have that $f^{-1}(\pi, M) = \pi'$ is a well-defined partition of A.

To show that π is a Nash stable and envy-free partition of A, consider the following proof by contradiction. Suppose that π is not envy Free; that there is a pair of agents $n_i, n_j \in N$ such that $v_i(t^i) < \Sigma_{k \in t^j} v_i(k)$. Consider that the corresponding agents

$p_i, p_j \in P$ will have the property that $u_i(r^i, c^i) < u_i(r^j, c^j)$. This contradicts the premise that (π, M) is an envy free solution of $f(A)$. Now suppose that π is not Nash stable; that there is an agent $n_i \in N$ and a team $t' \in \pi'$ such that $v_i(t' \cup \{n_i\}) > v_i(t^i)$. Consider that the corresponding agent $p_i \in P$ will have $u_i(r^i, c' \cup \{r^i\}) > u_i(r^i, c^i)$. This contradicts the premise that (π, M) is a Nash stable solution of $f(A)$. Therefore if $f(A)$ is in EF NS RBHG then A is in EF NS ASHG.

Now let π be a Nash stable partition of A. Let $(\pi', M) = f(\pi)$ be the corresponding partition in $f(A)$. For each agent $p_i \in P$, $u_i(r^i, c^i) = v_i(t^i)$ where $t^i \in \pi$ is composed of the agents represented by the roles in c^i.

By the same argument as in the previous case, we get that (π', M) is also envy free and Nash stable. Therefore if A is in EF NS ASHG then $f(A)$ is in EF NS RBHG.

Therefore $f(A)$ is in EF NS RBHG iff A is in EF NS ASHG. Thus, f is a reduction from EF NS ASHG to EF NS RBHG.

❏

Definition 4.2.6. *The language* IS RBHG *consists of those instances of RBHG for which there exists an individually stable solution.*

Theorem 4.2.7. IS RBHG *is NP-complete.*

Proof. To see that IS RBHG is in NP, observe that individual stability for a single RBHG agent can be verified in time $O(|P| \cdot |R|)$. Verifying the property for all agents requires time $O(|P|^2 \cdot |R|)$. To verify individual stability for a single agent, check if the agent can improve utility by moving from its current team to another team (with any of the possible roles) without decreasing utility for the agents already on the new team. In the worst case, where every agent is on a team by itself, this requires consideration of each of the $|P|$ possible teams and each of the $|R|$ possible roles on each team. Verifying the property for all agents requires performing the above test for each of the $|P|$ agents.

Next, we construct a reduction, f, from IS ASHG to IS RBHG. Let A be an instance of IS ASHG with a population N and utility function vector V. We use the same function f as in Theorem 4.2.2 to generate an instance $f(A) = (P, R, C, U)$ of RBHG.

Let $f(A)$ be in IS RBHG and let (π, M) be an individually stable solution of $f(A)$. From (π, M) we construct a partition π' of A using f^{-1}. Since (π, M) maps each $p_i \in P$ to the role r^i representing $f^{-1}(p_i) = n_i \in N$, we have that $f^{-1}(\pi, M) = \pi'$ is a well-defined partition of A.

To show that π is an individually stable partition of A, consider the following proof by contradiction. Suppose there were an agent $n_i \in N$ on a team $t^i \in \pi'$ and a team $t' \neq t^i \in \pi'$ such that $v_i(t' \cup \{n_i\}) > v_i(t^i)$ and for all $n_j \neq n_i \in t^i$ we have that $v_j(t^i) \leq v_j(t^i - \{n_i\})$. Consider that for the corresponding agent $p_i \in P$ in a team $t^i \in \pi$ with composition c^i there will be a team $t' \neq t^i \in \pi$ with composition c' such that $u_i(r^i, c' \cup \{r^i\}) > u_i(r^i, c^i)$ and for all $p_j \neq p_i \in t^i$ we have that $u_j(r^j, c^i) \leq u_j(r^j, c^j - \{r^i\})$. This contradicts the premise that (π, M) is an individually stable solution of $f(A)$. Therefore if $f(A)$ is in IS RBHG then A is in IS ASHG.

Now let π be an individually stable partition of A. Let $(\pi', M) = f(\pi)$ be the corresponding partition in $f(A)$. For each agent $p_i \in P$, $u_i(r^i, c^i) = v_i(t^i)$ where $t^i \in \pi$ is composed of the agents represented by the roles in c^i.

By the same argument as in the previous case, we get that (π', M) is also individually stable. Therefore if A is in IS ASHG then $f(A)$ is in IS RBHG.

Therefore $f(A)$ is in IS RBHG iff A is in IS ASHG. Thus, f is a reduction from IS ASHG to IS RBHG.

❑

Definition 4.2.8. *The language* CS RBHG *consists of those instances of RBHG* *for which there exists a non-empty core stable solution.*

Theorem 4.2.9. CS RBHG *is NP-complete.*

Proof. The construction given in the proof of Theorem 4.2.2 also gives a reduction from CS ASHG to CS RBHG.

Let $f(A)$ be in CS RBHG and let (π, M) be a core stable solution of $f(A)$. From (π, M) we construct a partition π' of A using f^{-1}. Since (π, M) maps each $p_i \in P$ to the role r^i representing $f^{-1}(p_i) = n_i \in N$, we have that $f^{-1}(\pi, M) = \pi'$ is a well-defined partition of A.

To show that π is a core stable partition of A, consider the following proof by contradiction. Suppose there were a subset of agents $N' \subset N$ such that, for each $n_i \in N'$, $v_i(N') > v_i(t^i)$. Consider that the corresponding subset of agents $P' \subset P$ such that, for each $p_i \in P'$, $u_i(r^i, c') > u_i(r^i, c^i)$ where $c' \subset R$ is composed of the roles represented by the agents in P'. This contradicts the premise that (π, M) is a core stable solution of $f(A)$. Therefore if $f(A)$ is in CS RBHG then A is in CS ASHG.

Now let π be a core stable partition of A. Let $(\pi', M) = f(\pi)$ be the corresponding solution of $f(A)$. For each agent $p_i \in P$, $u_i(r^i, c^i) = v_i(t^i)$ where $t^i \in \pi$ is composed of the agents represented by the roles in c^i. By the same argument as in the previous case, we get that (π', M) is also core stable. Therefore if A is in CS ASHG then $f(A)$ is in CS RBHG.

Therefore $f(A)$ is in CS RBHG iff A is in CS ASHG. Thus, f is a reduction from CS ASHG to CS RBHG.

❏

Definition 4.2.10. *The language* SCS RBHG *consists of those instances of RBHG for which there exists a non-empty strict core stable solution.*

Theorem 4.2.11. SCS RBHG *is NP-complete.*

Proof. The construction given in the proof of Theorem 4.2.2 also gives a reduction from SCS ASHG to SCS RBHG.

Let $f(A)$ be in SCS RBHG and let (π, M) be a strict core stable solution of $f(A)$. From (π, M) we construct a partition π' of A using f^{-1}. Since (π, M) maps each $p_i \in P$ to the role r^i representing $f^{-1}(p_i) = n_i \in N$, we have that $f^{-1}(\pi, M) = \pi'$ is a well-defined partition of A.

To show that π is a strict core stable partition of A, consider the following proof by contradiction. Suppose there were a subset of agents $N' \subset N$ such that, for each $n_i \in N'$, $v_i(N') \geq v_i(t^i)$ and $v_j(N') > v_j(t^j)$ for at least one $n_j \in N'$. Consider that the corresponding subset of agents $P' \subset P$ such that, for each $p_i \in P'$, $u_i(r^i, c') \geq u_i(r^i, c^i)$ and $u_j(r^j, c) > u_j(r^j, c^j)$ for at least one $p_j \in P'$, where $c' \subset R$ is composed of the roles represented by the agents in P'. This contradicts the premise that (π, M) is a strict core stable solution of $f(A)$. Therefore if $f(A)$ is in SCS RBHG then A is in SCS ASHG.

Now let π be a strict core stable partition of A. Let $(\pi', M) = f(\pi)$ be the corresponding solution of $f(A)$. For each agent $p_i \in P$, $u_i(r^i, c^i) = v_i(t^i)$ where $t^i \in \pi$ is composed of the agents represented by the roles in c^i. By the same argument as in the previous case, we get that (π', M) is also strict core stable. Therefore if A is in SCS ASHG then $f(A)$ is in SCS RBHG.

Therefore $f(A)$ is in SCS RBHG iff A is in SCS ASHG. Thus, f is a reduction from SCS ASHG to SCS RBHG.

❑

Definition 4.2.12. *The* grand coalition *for RBHG is a partition π of all agents to a single team t. In RBHG, there are several possible grand coalitions with different distributions of roles.*

Definition 4.2.13. *The language* GRAND PO RBHG *consists of those instances of RBHG for which there exists a partition consisting of the* grand coalition *and some* *assignment of agents to roles that is Pareto optimal.*

Theorem 4.2.14. GRAND PO RBHG *is coNP-complete.*

Proof. First we show that GRAND PO RBHG is in coNP. Given two solutions (π, M) and (π', M') for an instance of RBHG, we can check in polynomial time if (π', M') is a solution such that $u_i(r'^i, c'^i) \geq u_i(r^i, c^i)$ for all $p_i \in P$ and $u_j(r'^j, c'^j) > u_j(r^j, c^j)$ for at least one $p_j \in P$. Thus, given an instance $f(A)$ of RBHG and a solution (π, M) consisting of the grand coalition, it is NP to decide that π is *not* Pareto optimal.

The construction given in the proof of Theorem 4.2.2 also gives a reduction from GRAND PO ASHG to GRAND PO RBHG.

Let $f(A)$ be in GRAND PO RBHG and let (π, M) be a Pareto optimal solution of $f(A)$ consisting of the *grand coalition*. Observe that each agent $p_i \in P$ must be assigned to $r^i \in R$, or else (π, M) could not be Pareto optimal.

Observation 4.2.15. *By the construction, each agent $p_i \in P$ has $u_i(r, c) = -MaxAbsValue(A) \cdot |P| - 1$ when $r \neq r^i$. Since (π, M) is a Pareto optimal solution, each agent $p_i \in \pi$ must be matched to $r^i \in R$. Otherwise the partition could be improved by assigning each agent p_i to its role r^i.*

Therefore if a partition π consisting of the grand coalition *is Pareto optimal, then $M(p_i) = r^i$ for each agent $p_i \in P$.*

From (π, M), a Pareto optimal solution of $f(A)$ consisting of the *grand coalition*, we construct a partition π of A using f^{-1}. Since (π, M) maps each $p_i \in P$ to the role r^i representing $f^{-1}(p_i) = n_i \in N$, we have that $f^{-1}(\pi, M) = \pi$ is a well-defined partition of A consisting of the *grand coalition*.

To show that π is a Pareto optimal partition of A, consider the following proof by contradiction. Suppose there were a partition π'_δ of A such that, for each $n_i \in N$ assigned to its team $t^i_\delta \in \pi'_\delta$, $v_i(t^i_\delta) \geq v_i(N)$ and for at least one $n_j \in N$ assigned to its team $t^j_\delta \in \pi_delta'$, $v_j(t^j_\delta) > v_i(N)$. Consider that the corresponding a solution $f(\pi'_\delta, M'_\delta)$ of $f(A)$ such that each agent p_i is assigned to r^i and, for each $p_i \in P$, $u_i(r^i, c^i_\delta) \geq u_i(r^i, c^i)$ and, for at least one $p_j \in P$, $u_j(r^j, c^j_\delta) > u_i(r^j, c^j)$. This contradicts the premise that (π, M) is a Pareto optimal solution for $f(A)$. Therefore if $f(A)$ is in GRAND PO RBHG then A is in GRAND PO ASHG.

Now let π be a Pareto optimal partition of A consisting of the *grand coalition.* Let $(\pi', M') = f(\pi)$ be the corresponding solution of $f(A)$ consisting of the *grand coalition.* For each agent $p_i \in P$, $u_i(r^i, c^i) = v_i(t^i)$ where $t^i \in \pi$ is composed of agents represented by roles in c^i. By the same argument as in the previous case, we get that (π', M') is also Pareto optimal. Therefore if A is in GRAND PO ASHG then $f(A)$ is in GRAND PO RTHG.

Therefore $f(A)$ is in GRAND PO RBHG iff A is in GRAND PO ASHG. Thus, f is a reduction from GRAND PO ASHG to GRAND PO RBHG. ❑

Definition 4.2.16. *The language* GRAND CSCS RBHG *consists of those instances of RBHG for which there exists a partition consisting of the* grand coalition *and some assignment of agents to roles which is contractual strict core stable.*

Theorem 4.2.17. GRAND CSCS RBHG *is coNP-complete.*

Proof. First we show that GRAND CSCS RBHG is in coNP by the following nondeterministic polynomial time algorithm. Given a solution (π, M) consisting of the *grand coalition* for an instance of RBHG, guess a team t' and a composition c'. If (t', c') *weakly blocks* (π, M) for some matching of roles in c' to agents in t', then (π, M) is *not* contractual strict core stable.

The construction given in the proof of Theorem 4.2.2 also gives a reduction from GRAND CSCS ASHG to GRAND CSCS RBHG.

Let $f(A)$ be in GRAND CSCS RBHG and let (π, M) be a contractual strict core stable solution of $f(A)$ consisting of the *grand coalition.* Observe that each agent in $p_i \in \pi$ must be assigned to $r^i \in R$, or else π could not be Pareto optimal. This follows from Observation 4.2.15.

From (π, M), a contractual strict core stable solution of $f(A)$ consisting of the *grand coalition*, we construct a partition π' of A using f^{-1}. Since (π, M) maps each $p_i \in P$ to the role r^i representing $f^{-1}(p_i) = n_i \in N$, we have that $f^{-1}(\pi, M) = \pi'$ is a well-defined partition of A consisting of the *grand coalition*.

To show that π is a contractual strict core stable partition of A, consider the following proof by contradiction. Suppose π' of A is not contractual strict core stable. Then there there exists a *weakly blocking* team t' of π'. Consider that the corresponding team t' and composition c' which *weakly blocks* $f(\pi, M)$, where the composition c' consists of roles for agents in t' and each agent $p_i \in t'$ is matched to its own role $r_i \in c'$. This contradicts the premise that (π, M) is Pareto optimal. Therefore if $f(A)$ is in GRAND CSCS RBHG then A is in GRAND CSCS ASHG.

Now let π be a contractual strict core stable partition of A consisting of the *grand coalition*. Let $(\pi', M') = f(\pi)$ be the corresponding solution of $f(A)$ consisting of the *grand coalition*. For each agent $p_i \in P$, $u_i(r^i, c^i) = v_i(t^i)$ where $t^i \in \pi$ is composed of agents represented by roles in c^i. By the same argument as in the previous case, we get that (π', M') is also contractual strict core stable. Therefore if A is in GRAND CSCS ASHG then $f(A)$ is in GRAND CSCS RTHG.

Therefore $f(A)$ is in GRAND CSCS RBHG iff A is in GRAND CSCS ASHG. Thus, f is a reduction from GRAND CSCS ASHG to GRAND CSCS RBHG.

❑

Chapter 5 Heuristic Matchmaking Results

In this Chapter I present several experimental procedures and the results of their application to real world matchmaking scenarios. The main research question of this chapter is whether or not the algorithms I propose can produce partitions with better optimization results than partitioning without intent to optimize.

In Section 5.3 I outline a procedure I have used to scrape real world data from League of Legends matches. I detail a few observations about the structure of the real world data and propose two different utility functions for converting the matchmaking data into RBHG instances. In Section 5.4 I detail four methods of generating randomized RBHG instances to test extreme cases of observed properties in the League of Legends data: the probability distribution over roles and compositions, the presence of a single role which is popular to all agents, the presence of a single role which is unpopular to all agents (yet still required for several compositions they do prefer), and the phenomenon of each agent having devotion to a single role it prefers.

In Sections 5.1 and 5.2, I provide implementations of two heuristic matchmaking procedures which I call *greedy voting heuristic* and *greedy local search heuristic.* The greedy voting heuristic iteratively forms new teams by selecting a single composition to optimize on. The local search heuristic instead begins with a single agent and iteratively adds agents to the team which provide the most improvement to the total utility of the optimal matching for that team. Both algorithms run in polynomial time in the size of the input.

I evaluate the two algorithms on RBHG instances generated from the League of Legends matchmaking data and the four experimental utility functions for randomly generated RBHG instances. I compare the results of the algorithms to randomly generated partitions on multiple optimization goals in Section 5.5.

Greedy local search shows general improvement over greedy voting and random partitions. The instances generated from the League of Legends data and on two of the four experimental utility functions tend to produce fewer reachable compositions in any given partition. This is especially true for partitions formed by the greedy voting heuristic.

5.1 Greedy Voting Heuristic

A *voting rule* is a function mapping a vector a of voters' votes to one of the b candidates in a candidate set c. With the *scoring* voting rule, each voter assigns a score to each candidate within the set. The winner of the election is the candidate with the most total points over all voters in the election.

Definition 5.1.1. *[24] Let $a = \langle a_1, \cdots, a_m \rangle$ be a vector of integers such that $a_1 < a_2 < \ldots < a_m$. For each voter, a candidate receives a_1 points if it is ranked first by the voter, a_2 points if it is ranked second, etc. The score s_c of candidate c is the total number of points the candidate receives. This is a* scoring *voting rule.*

By modeling agents as voters in an election and their preferences over team compositions and roles as votes, the scoring voting rule can be applied to iteratively hold elections over the set of compositions. Once a composition is selected, the set of agents who optimize total utility for that composition can be assigned to a team together. This greedy approach narrows the problem to optimizing utility for a single composition rather than across all compositions.

For my procedure called GREEDY VOTING HEURISTIC, the voters are the agents and the candidates are pairs (c, r) where c is a composition and $r \in c$. An election is run upon the candidate set to select the most-preferred composition c_i. A set of $|c_i|$ voters with the highest utility for that composition is selected to form a team and removed from the population. These agents are matched to roles within the composition by application of the Kuhn Munkres algorithm for optimizing square matrices [31]. This optimizes total utility of the matching for players on the new team. Their votes are removed, and a new election is held on the reduced candidate set. This procedure continues until all $|P|$ agents have been matched to teams. The following pseudocode describes this greedy algorithm [69]:

Algorithm 1 GreedyVoting(RTHG instance G, empty partition π, $\textsc{min}(|c|) = m$)

```
$b =: |C|$
for $|C|$ compositions $c_0 \to c_b$ do
  for $|c_i|$ positions $r_0 \to r_{|c_i|} \in c_i$ do
    calculate the sum of agent votes on $\langle c_i, r_j \rangle$. %$\mathcal{O}(|P|)$
  end for
end for
for $|P|/m$ coalitions $t_0 \to t_{|P|/m-1}$ to assign to $\pi$ do
  find the set of compositions $C_{max}$ for which the sum of total votes is maximized.
  %$\mathcal{O}(|C| \cdot m)$
  select one composition $c_i$ uniformly at random from within the set.
  for $|c_i|$ positions $r_0 \to r_{|c_i|-1} \in c_i$ do
    find the set of agents $P_{max}(c_i, r_j)$ for whom the individual agent's vote for
    $\langle c_i, r_j \rangle$ is maximized. %This takes time $\mathcal{O}(|P|/|c_i|)$, given that the population
    shrinks by $|c_i|$ agents as each team is formed and removed.
    select one agent $p_j$ uniformly at random from within the set.
    add agent $p_j$ to the coalition $t_k$.
    optimize the $|c_i| \times |c_i|$ matrix of agent utilities over the roles in composition $c_i$
    by the Kuhn Munkres algorithm.
    for $|C|$ compositions $c_0 \to c_{|C|-1}$ do
      for $m$ positions $r_0 \to r_{m-1} \in c_i$ do
        remove agent $p_j$'s vote from the population, decrementing the sum total
        vote on $\langle c_i, r_j \rangle$.
      end for
    end for
  end for
  append team $t_k$ to the partition $\pi$.
end for
```

Observation 5.1.2. *The time complexity of* greedy voting heuristic *is* $\mathcal{O}(|P|^2 \cdot m^2)$.

5.2 Greedy Local Search Heuristic

In this section, I define a greedy local search heuristic which runs in polynomial time. I prove that, for a special case of RBHG, this algorithm will always produce a perfect assignment.

The greedy local search heuristic I introduce for RBHG iteratively selects an agent $p \in P$ as the *pivotal agent* and locally optimizes either *expected utility* or *expected stability* for the team t including p and some new agent $p' \in P$. The local search continues to add agents to the team t until no local improvement is available or all positions have been filled, at which point t is added to the final partition. A new pivotal agent is then selected, forming a new team around the pivot. This procedure is repeated for each of the up to $|P|/\textsc{max}(|t|)$ teams. In this procedure, a *pivotal agent* must be selected each iteration. I consider three different methods for selecting the pivotal agent, which I will detail in Section 5.5. Each of three methods I use allows for the selection of a pivotal agent to be made in constant time. The pseudocode for the greedy algorithm follows:

Algorithm 2 GreedyLocalSearch(RBHG instance B, empty partition π)

```
for ($|P|/\textsc{min}(|c|)$) teams do
    select a pivotal agent $p$
    for $\textsc{max}(|c|) - 1$ positions do
        set max index $i_{\textsc{max}}$ to null
        set max score $s_{\textsc{max}}$ to $\textsc{min}(u_p(r,c)) \cdot |P|$
        for each $p'$ of the $|P|/m$ remaining agents do
            calculate expected utility $s'_p$ for $t \cup p'$ %$\mathcal{O}(|C| \cdot \textsc{max}(|c|)^3)$
            if $s'_p > s_{\textsc{max}}$, set $i_{\textsc{max}} = p'$ and $s_{\textsc{max}} = s'_p$
        end for
        set $t = t \cup i_{\textsc{max}}$
    end for
    set $\pi = \pi \cup t$
end for
```

Observation 5.2.1. *The time complexity of* greedy local search heursitic *is* $\mathcal{O}(|P|^2 \cdot |C| \cdot \textsc{max}(|c|)^2)$.

5.3 Scraping League of Legends Game Data

For my experiments, I consider a population of League of Legends players where $|P| = 1081$. I scraped data from the most recent 20 matches for each player $p \in P$ from the League of Legends game statistics website *LolKing* [21], maintained by ZAM Network LLC. I stored the roles, compositions, and win/loss records for each match. I selected players uniformly at random by generating account numbers from 20,000,000 to 60,000,000. This gives a range of accounts having been created roughly between the years 2012 and 2014. After generating a number, I checked if the account was active by checking the dates of the most recent 10 matches. Players whose most recent 10 matches were played within the last 7 days and had a team size of 5 were accepted for scraping, while others were rejected. Among those accepted, I checked again in 7 days and scraped for additional match data. I accepted those whose matches continued to meet my criteria and rejected the rest. This procedure left me with the population of 1081 players to consider.

I considered a set R with $|R| = 5$ consisting of popular champion roles, *Jungler, AD Carry, Tank, Support,* and *Mage.* These roles were identified as the most common roles of champions selected. To handle multiple identical roles within the same composition, I accumulated a counter to create multiple instances of the same role. I used frequent item set mining over the sets of roles used each match to determine the most frequently played compositions within the population. Frequent item set mining was performed using a *recursive elimination* implementation by Borgelt [17]. I used a *support threshold* of 3%; that is, I accepted those compositions which were used (i.e. *supported*) in at least 3% of matches, and rejected others. This left a set $|C| = 8$ of compositions which in total were used in more than 60% of all matches played in the games I considered.

In Table 5.1 I show the percentage of winning teams which used each composition within the support threshold. Table 5.2 presents the percentage of losing teams

Table 5.1: Percent usage of compositions by winning teams, support threshold $\geq 3\%$

Composition	**Percent Usage (%)**
J1 S1 T1 M1 A1	(20.3255)
M2 S1 T1 A1 M1	(9.312)
T2 S1 M1 A1 T1	(7.25911)
M2 J1 T1 A1 M1	(6.15868)
M2 J1 S1 A1 M1	(5.61309)
J2 S1 M1 A1 J1	(5.16923)
T2 M2 A1 T1 M1	(3.42149)
T2 J1 M1 A1 T1	(3.32902)

Table 5.2: Percent usage of compositions by losing teams, support threshold $\geq 3\%$

Composition	**Percent Usage (%)**
J1 S1 T1 M1 A1	(19.2343)
M2 S1 T1 A1 M1	(7.58276)
M2 S1 J1 A1 M1	(6.64879)
M2 J1 T1 A1 M1	(6.50083)
T2 S1 M1 A1 T1	(5.35417)
T2 J1 M1 A1 T1	(4.62364)
J2 S1 M1 A1 J1	(4.2445)
T2 M2 A1 T1 M1	(3.02386)

which used each composition. Roles within compositions are numbered to distinguish multiple instances of the same role. Table 5.3 presents the percent likelihood that a given agent will accept one of the five popular roles. The numbering here shows the likelihood that an agent will accept being the first, second, or even third instance of the given role within a composition. Notably, it is more likely for a third *Mage* to be adopted onto a team than even a second *AD Carry*, even though the *AD Carry* is the most popular role of all. For players familiar with the game, this may not be surprising as the *AD Carry* role tends to require more of the limited team resources in order to do well. As such, having a second or much less third person fulfilling that role within a team is not common practice. I observe from the data that most people want to play this position but few people want there to be two of them.

Table 5.3: Percent usage of roles across all teams, support threshold $\geq 3\%$

Role	**Percent Usage (%)**
A1	(90.836)
M1	(89.1807)
T1	(76.6969)
S1	(68.2264)
J1	(64.8696)
M2	(41.1873)
T2	(25.7259)
J2	(14.9898)
M3	(9.44146)
S2	(6.82449)
A2	(5.91825)
T3	(4.30923)

5.4 Randomly Generated RBHG Instances

I generated 80 RBHG instances, each with population size $|P| = 1081$, to compare with the heuristic result on two instances generated from League of Legends matchmaking data. I used the set of eight compositions and five roles from the scraped League of Legends data. I used four utility distributions, each over 20 of the RBHG instances: *League of Legends based utilities*, *one popular role*, *one unpopular role*, and *devoted role* utilities. Primarily, the question being asked with each experiment is whether or not the proposed utility distribution produces similar results to the real world data. Secondarily, I am examining the algorithms' performances with respect to *expected stability*, *expected utility*, and *optimal composition utility* with respect to these distributions.

Utility for each role and composition was set to a default value of 0. For each agent, 10 simulated matches were generated assigning the agent to a role and composition. Utility for the agent on these roles and compositions was set to 1. For each simulated match, the random number generator from the Python *random* library was used to decide which role and composition to select.

League of Legends based utilities: A probability distribution over the compositions and roles was created. When generating a match for an agent, the probability of the agent accepting a particular role and composition was equal to the probability of an agent accepting the same role and composition in the League of Legends data. The resulting utility distribution is similar to what is observed in the League of Legends matches but does not guarantee a modeling of individual agent behaviors. The question being asked by this experiment is whether the population-wide probability distribution over role and composition pair acceptance is sufficient to predict similar matchmaking results to the real world data.

One popular role: In each instance, a single role was selected to be most popular to all agents. This is analogous to the observed preference of League of Legends players to play the *AD Carry* (Table 5.3), but taken to an extreme. When generating a preference for a role and composition pair, the probability of the agent selecting the popular role was 80%, with each of the other four roles being selected with 5% probability. The composition for each match was selected uniformly at random from the set. The question being asked by this experiment is whether or not having a single role popular to the entire population produces similar matchmaking results to the real world data.

One unpopular role: In each instance, a single role was selected to be least popular to all agents. This is analogous to the observed dislike of League of Legends players to fulfill the role of *Support* (Table 5.3), again to the extreme. When generating a preference for a role and composition pair, the probability of the agent selecting the unpopular role was 5%, with each of the other four roles being selected with 23.75% probability. The composition for each match was selected uniformly at random from the set. The question being asked by this experiment is whether or not having a single role unpopular to the entire population produces similar matchmaking results to the real world data.

Devoted role: In each instance, for each agent, a single role was selected to be strictly preferred by that agent. This is analogous to a League of Legends player who is devoted to selecting a single preferred role whenever possible. When generating a preference for a role and composition pair in these instances, an agent is always assigned to its preferred role if it is available in that composition. If the preferred role is not available in the composition, a role is selected uniformly at random. The composition for each match was selected uniformly at random from the set. The question being asked by this experiment is whether or not having each agent devote themselves to a single role produces similar matchmaking results to the real world data.

5.5 Testing and Results

As previously observed, *optimal composition utility*, *expected utility* and *expected stability* can be calculated in polynomial time given a partition π. The challenge is selecting a partition of agents to teams for which these values are optimized. I evaluate the partitions generated by each algorithm in terms of the maximum, minimum, mean and median for each of *optimal composition utility*, *expected utility* and *expected stability.* I consider *expected stablity* and *expected utility* both over the number of *reachable compositions* and over the number of *reachable* and *advocated compositions.*

The choice of pivotal agent for *greedy local search heuristic* can impact the results of the algorithm. I considered three methods of selecting a pivotal agent: selecting an agent who is among the *easiest to please*, an agent who is among the *hardest to please*, and an agent selected uniformly at random.

Definition 5.5.1. *Given an instance B of RBHG, an agent p_i who is among the* easiest to please *is one for whom $q_i = \sum_{r,c} u_i(r,c)$ is maximized. An agent who is among the* hardest to please *is one for whom q_i is minimized.*

I ran *greedy local search heuristic* upon the entire population $|P| = 1081$ generated from the League of Legends game data and on the 20 randomly generated instances. I tested three variants of Greedy Local Search, varying the method of selecting the pivotal agent: selecting one of the *easiest to please* agents each iteration, one of the *hardest to please* agents, and an agent selected i.i.d. from P.

I ran *greedy voting heuristic* upon the same population. In addition, I formed 100 partitions of agents to teams selected i.i.d. This comparison tests how the two algorithms fare against matchmaking without attempting optimization.

For the League of Legends data I tested two utility functions, *wins only* and *wins and losses*, each generated from the roles and compositions with which agents won or lost in the game data considered. In both cases, I consider an agent to *accept* a pair $\langle r, c\rangle$ if $u_p(r,c) = 1$ and otherwise to *reject* it.

Definition 5.5.2. Wins only *utility function: For each agent $p \in P$, for each $c \in C$ set $u_p(r,c) = 1$ if the agent won more matches with $\langle r, c\rangle$ than they lost, and $u_p(r,c) = 0$ otherwise.*

Definition 5.5.3. Wins and losses *utility function: For each agent $p \in P$, for each $c \in C$ set $u_p(r,c) = 1$ if the agent won more matches with $\langle r, c\rangle$ than they lost, $u_p(r,c) = -1$ if the agent lost more matches with $\langle r, c\rangle$ than they won, and $u_p(r,c) = 0$ otherwise.*

Computations were run on a machine using 8 GB of RAM and a 2.50 GHz Intel(R) Core(TM) i5-3210M CPU. Each algorithm was implemented in Python 3.4.

5.5.1 Expected Stability Results

Experimental results in terms of *expected stability* are found in Figures 5.1–5.12. The column labeled *Max* denotes the maximum *expected stability* across all teams in the partition, while the column labeled *Min* denotes the minimum across all teams. The columns labeled *Mean* and *Median* denote the mean and median across all teams. For *expected stability* the maximum possible value is 1. Note that Figures 5.1, 5.3, 5.5, and 5.9 over reachable compositions use a smaller maximum value for the Y axis compared to Figures 5.2, 5.4, 5.6, and 5.10. This is due to the smaller magnitudes of their results.

The *greedy voting heuristic* shows minor improvement over random partitions with regards to maximum and mean expected stability for all utility functions. However, with only one exception (Figure 5.3), the *greedy local search heuristic* produces higher *expected stability* than both random partitions and the *greedy voting heuristic.* This result occurs regardless of the selection of pivotal agent each iteration of the local search. Instances generated from the scrapped League of Legends data had better results by selecting the *hardest to please* agent as the pivot, while selecting a random pivot tends to be the best choice in other instances.

Expected stability was difficult to optimize–especially in terms of the minimum across all teams. For all approaches, on all six utility functions tested, minimum expected stability was zero. The same is true of the medians for both the *one popular role* (Figures 5.3 and 5.4) and *one unpopular role* (Figures 5.5 and 5.6) generated instances. It is possible that there were no possible partitions for these instances with a higher minimum expected stability. Note that a partition with a minimum expected stability greater than zero contains a perfect matching, given that each team has at least one composition with a perfect matching. As previously observed, not every RBHG instance contains a perfect solution and it is NP-complete to decided if one exists for a given instance.

Best results were observed in the *devoted role* instances (Figures 5.11 and 5.12). Instances with *League of Legends based* utilities and those generated directly from League of Legends matchmaking data showed similar improvement, but not to the same magnitude (with the exception of a single team maintaining perfect expected stability even over *reachable compositions* in Figure 5.7). The solutions for these instances contained fewer reachable compositions on each team compared to the other experimental utility functions.

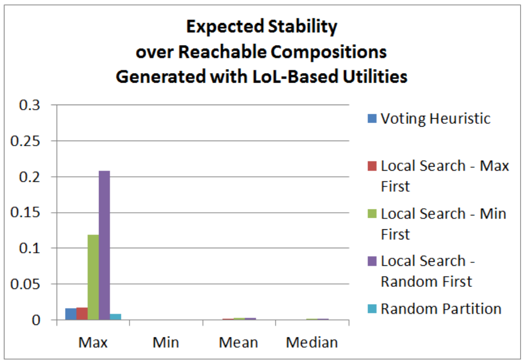

Figure 5.1: Expected stability over reachable compositions with generated League of Legends match data using utility function $u_p(r, c) \to (1, 0)$

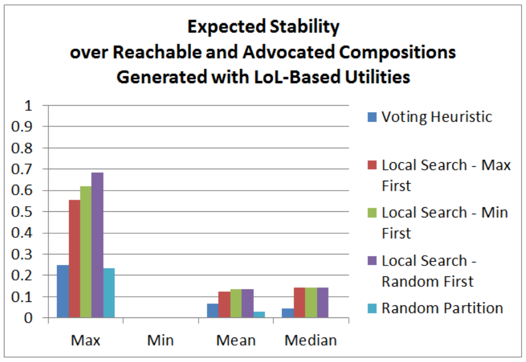

Figure 5.2: Expected stability over reachable and advocated compositions with generated League of Legends match data using utility function $u_p(r, c) \to (1, 0)$

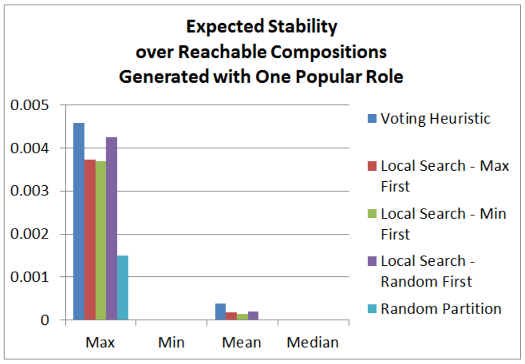

Figure 5.3: Expected stability over reachable compositions with generated *one popular role* match data using utility function $u_p(r, c) \to (1, 0)$

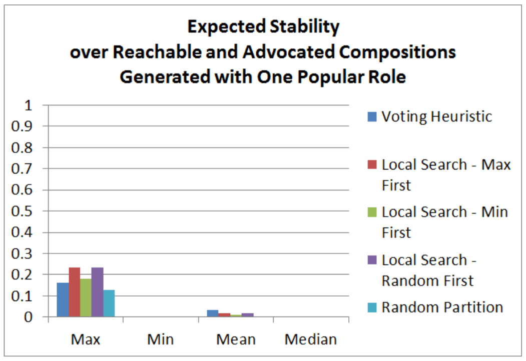

Figure 5.4: Expected stability over reachable and advocated compositions with generated *one popular role* match data using utility function $u_p(r, c) \to (1, 0)$

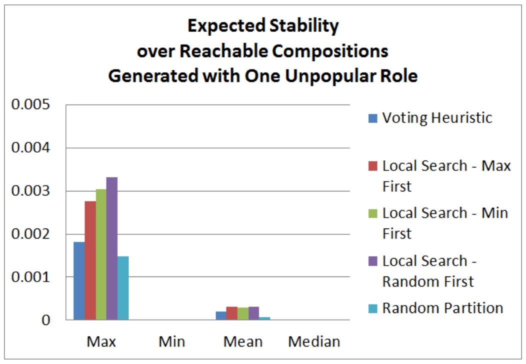

Figure 5.5: Expected stability over reachable compositions with generated *one unpopular role* match data using utility function $u_p(r, c) \to (1, 0)$

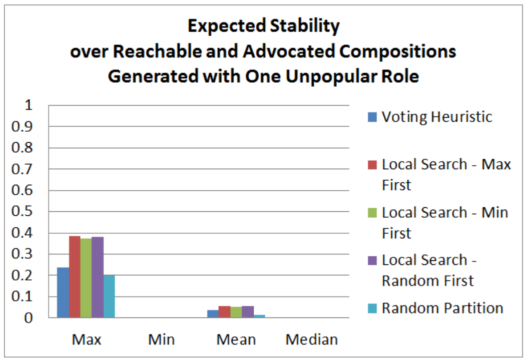

Figure 5.6: Expected stability over reachable and advocated compositions with generated *one unpopular role* match data using utility function $u_p(r, c) \to (1, 0)$

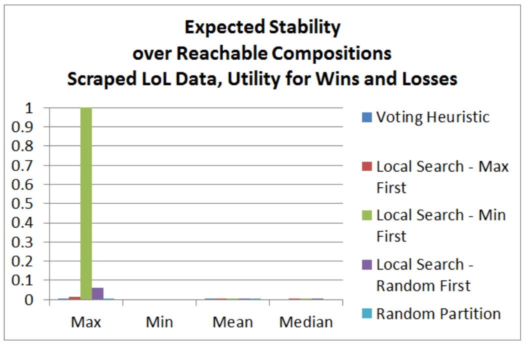

Figure 5.7: Expected Stability over reachable compositions with League of Legends match data using utility function $u_p(r, c) \to (1, 0, -1)$

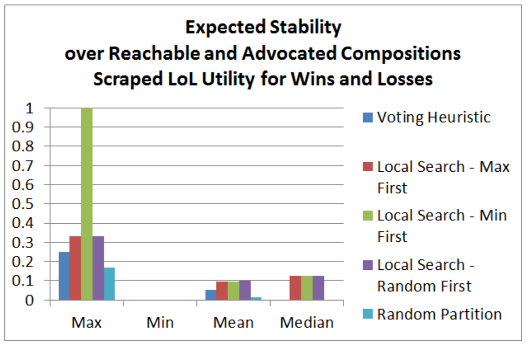

Figure 5.8: Expected Stability over reachable and advocated compositions with League of Legends match data using utility function $u_p(r, c) \to (1, 0, -1)$

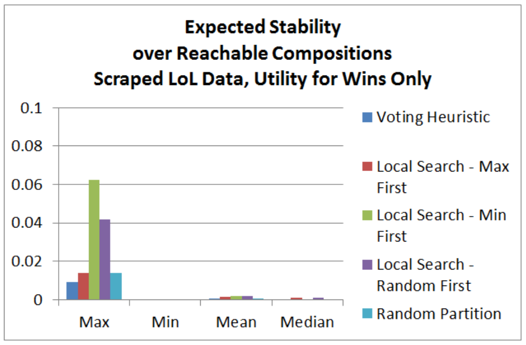

Figure 5.9: Expected stability over reachable compositions with League of Legends match data using utility function $u_p(r,c) \to (1,0)$

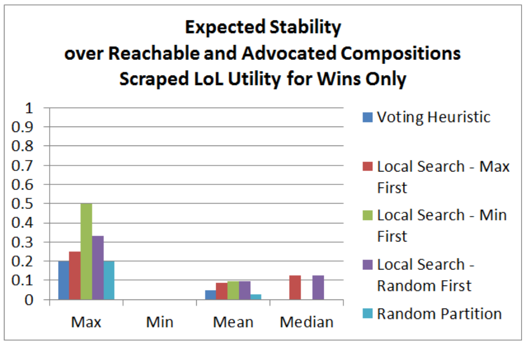

Figure 5.10: Expected stability over reachable and advocated compositions with League of Legends match data using utility function $u_p(r,c) \to (1,0)$

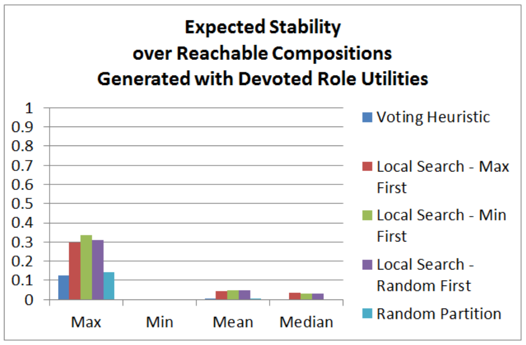

Figure 5.11: Expected stability over reachable compositions with generated *devoted role* match data using utility function $u_p(r,c) \to (1,0)$

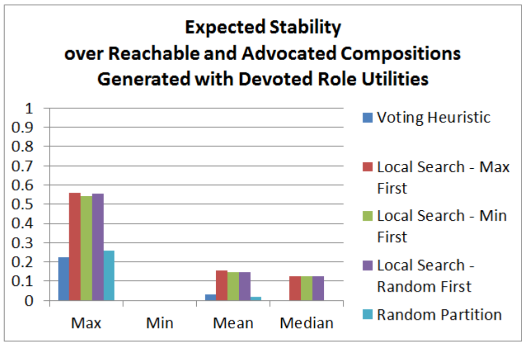

Figure 5.12: Expected stability over reachable and advocated compositions with generated *devoted role* match data using utility function $u_p(r,c) \to (1,0)$

5.5.2 Expected Utility Results

Experimental results in terms of *expected utility* are found in Figures 5.13–5.24. The column labeled *Max* denotes the maximum *expected utility* across all teams in the partition, while the column labeled *Min* denotes the minimum across all teams. The columns labeled *Mean* and *Median* denote the mean and median across all teams. For *expected utility* the maximum possible value is 5. Note that Figures 5.13, 5.15, 5.17, 5.19, and 5.21 over reachable compositions use a smaller maximum value for the Y axis compared to Figures 5.12, 5.14, 5.16, 5.18, and 5.20. This is due to the smaller magnitudes of their results.

The *greedy voting heuristic* does not demonstrate general improvement over random partitions with regards to maximum, mean and median expected utility. For min expected utility, random partitioning shows an improvement over *greedy voting heuristic*.

In one case (Figure 5.23), random partitioning showed improvement over one (but not all) of the the local search pivot selection methods. Otherwise, the *greedy local search heuristic* showed improvement for expected utility over random partitioning in all cases.

The *greedy voting heuristic* shows a large improvement over the *greedy local search* algorithm in terms of the maximum *expected utility* over *reachable compositions* on instances generated from League of Legends matchmaking data (Figures 5.19 and 5.21) and on instances generated with League of Legends based utility distributions (Figure 5.13). This is due to fact that, in these instances, the composition consisting of one of each role is highly popular. Because the *greedy voting heuristic* is designed to select a team based upon the most popular composition each iteration, it often selects a team which optimizes utility on this composition. This tends to lower the number of reachable compositions, with each agent having a preference for a unique role in at least this composition. As the improvement does not carry over to the mean and median, this does not necessarily suggest a general improvement by the *greedy voting heuristic*. Furthermore, the improvements do not hold up when only *reachable* and *acceptable compositions* are considered (Figures 5.14, 5.20, and 5.22).

For the mean and median, no strong improvements of *expected utility* over *reachable compositions* are observed for either the *greedy voting heuristic* or the *greedy local search heuristic* compared to random partitions. While I observe that the magnitude of results are once again higher overall for the *devoted role* utility function, the *greedy local search heuristic* only tends to improve on *expected utility* over *reachable* and *acceptable compositions*. While *expected stability* over *reachable compositions* was improved by the role/-devotion phenomenon, *expected utility* does not share this relationship on these instances.

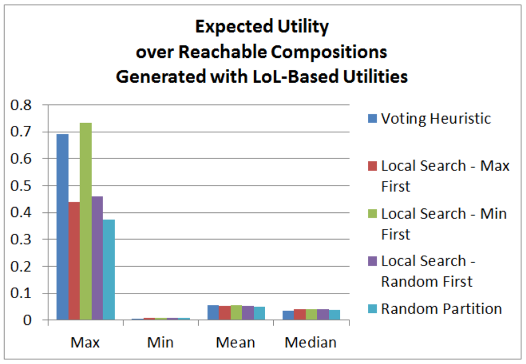

Figure 5.13: Expected utility over reachable compositions with generated League of Legends match data using utility function $u_p(r, c) \to (1, 0)$

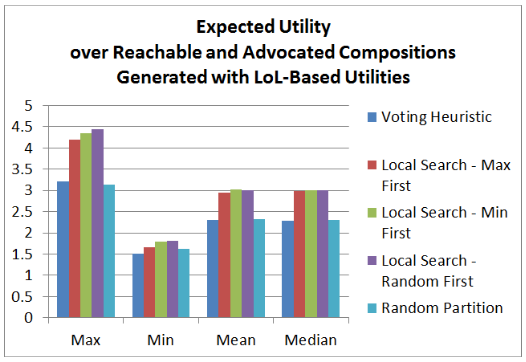

Figure 5.14: Expected utility over reachable and advocated compositions with generated League of Legends match data using utility function $u_p(r, c) \to (1, 0)$

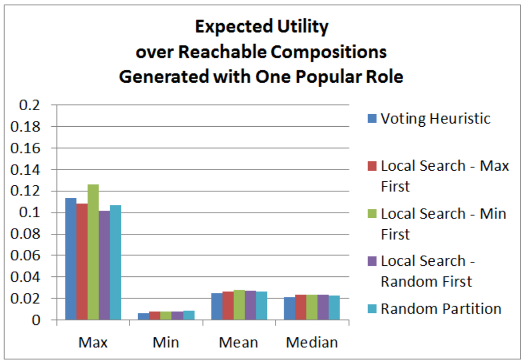

Figure 5.15: Expected utility over reachable compositions with generated *one popular role* match data using utility function $u_p(r, c) \to (1, 0)$

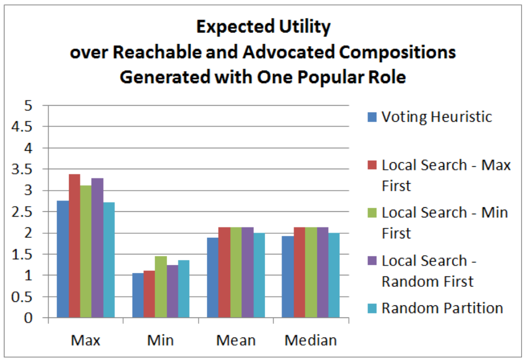

Figure 5.16: Expected utility over reachable and advocated compositions with generated *one popular role* match data using utility function $u_p(r, c) \to (1, 0)$

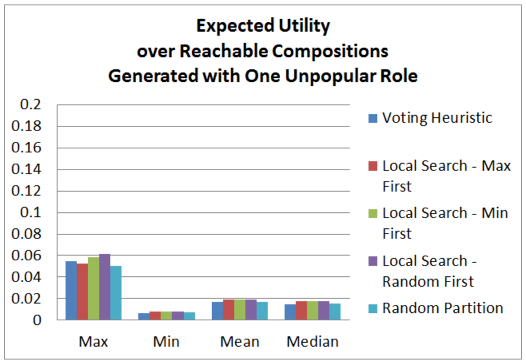

Figure 5.17: Expected utility over reachable compositions with generated *one unpopular role* match data using utility function $u_p(r, c) \to (1, 0)$

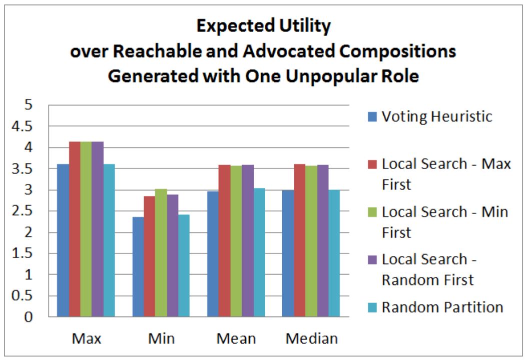

Figure 5.18: Expected utility over reachable and advocated compositions with generated *one unpopular role* match data using utility function $u_p(r, c) \to (1, 0)$

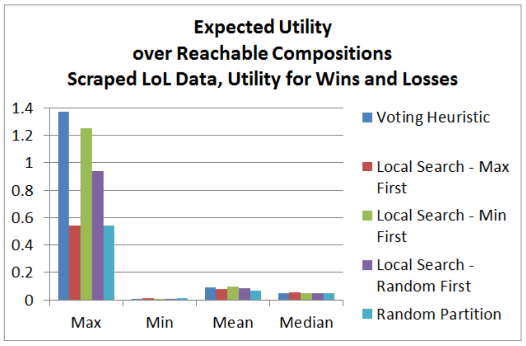

Figure 5.19: Expected utility over reachable compositions with League of Legends match data using utility function $u_p(r, c) \rightarrow (1, 0, -1)$

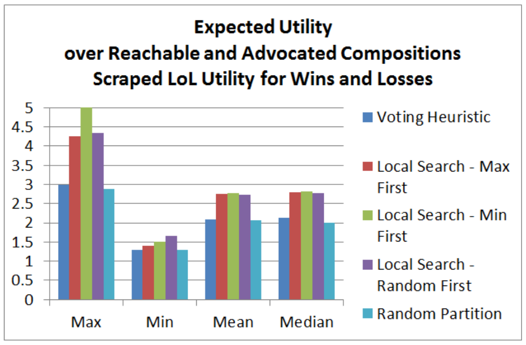

Figure 5.20: Expected utility over reachable and advocated compositions with League of Legends match data using utility function $u_p(r, c) \rightarrow (1, 0, -1)$

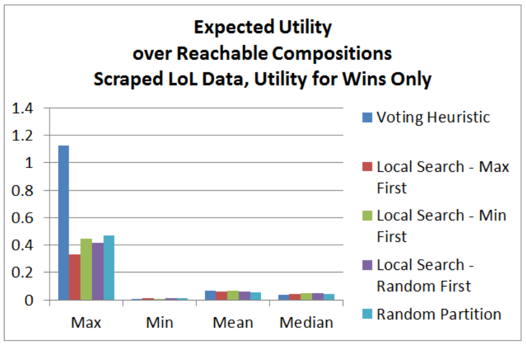

Figure 5.21: Expected utility over reachable compositions with League of Legends match data using utility function $u_p(r, c) \rightarrow (1, 0)$

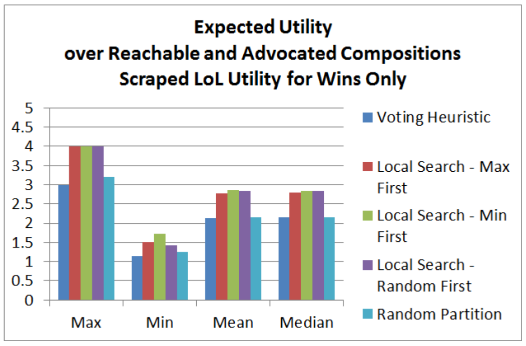

Figure 5.22: Expected utility over reachable and advocated compositions with League of Legends match data using utility function $u_p(r, c) \rightarrow (1, 0)$

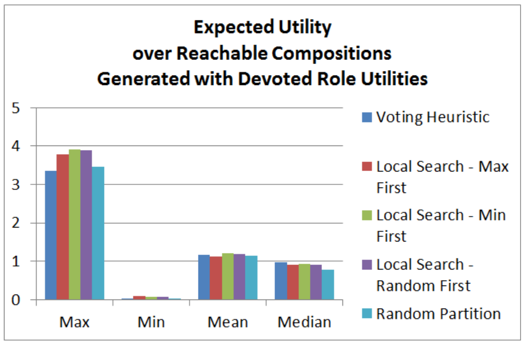

Figure 5.23: Expected utility over reachable compositions with generated *devoted role* match data using utility function $u_p(r, c) \rightarrow (1, 0)$

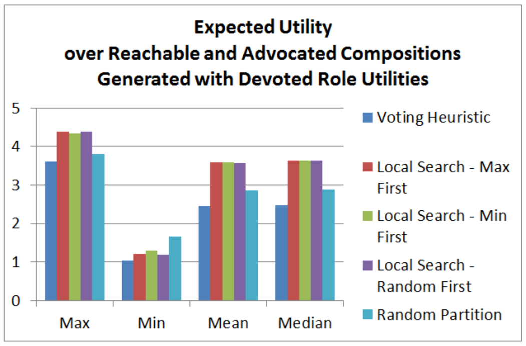

Figure 5.24: Expected utility over reachable and advocated compositions with generated *devoted role* match data using utility function $u_p(r, c) \rightarrow (1, 0)$

5.5.3 Optimal Composition Utility Results

Experimental results in terms of *optimal composition utility* are found in Figures 5.25–5.30. The column labeled *Max* denotes the maximum *optimal composition utility* across all teams in the partition, while the column labeled *Min* denotes the minimum across all teams. The columns labeled *Mean* and *Median* denote the mean and median across all teams. For *optimal composition utility* the maximum possible value is five.

It is worth noting that *optimal composition utility* is the only metric that the *greedy voting heuristic* was designed to improve. Rather than attempting to optimize over all compositions, the voting heuristic works by selecting a highly preferred composition and then matching agents to roles within that composition in an optimal way. Therefore, I would expect the voting heuristic to perform better on this test case than on the previous two.

None of the methods seem to have trouble optimizing the maximum *optimal composition utility* among teams. It seems that even with a random partition it is likely to find at least one team with a perfect matching for each of the five utility distributions considered.

For the minimum optimal composition utility, the *greedy local search heuristic* performs as well as or better than both the *greedy voting heuristic* and random partitioning on all cases except on the *devoted role* instances (Figure 5.30). Similarly, with only one exception (Figure 5.26), the *greedy local search heuristic* shows improvement for both the mean and the median on these instances.

Though it is generally beaten by local search, the *greedy voting heuristic* performs as well as or better than random partitions in terms of the mean and median. This is in contrast to the lack of significant improvement seen from the voting heuristic for *expected stability* and *expected utility.* In the case of utility distributions with one popular role (Figure 5.26) the voting heuristic actually out-performs the local search heuristic in terms of the mean. While the local search heuristic generally shows improvement over the voting heuristic, the difference is less sweeping and pronounced for the optimal composition utility.

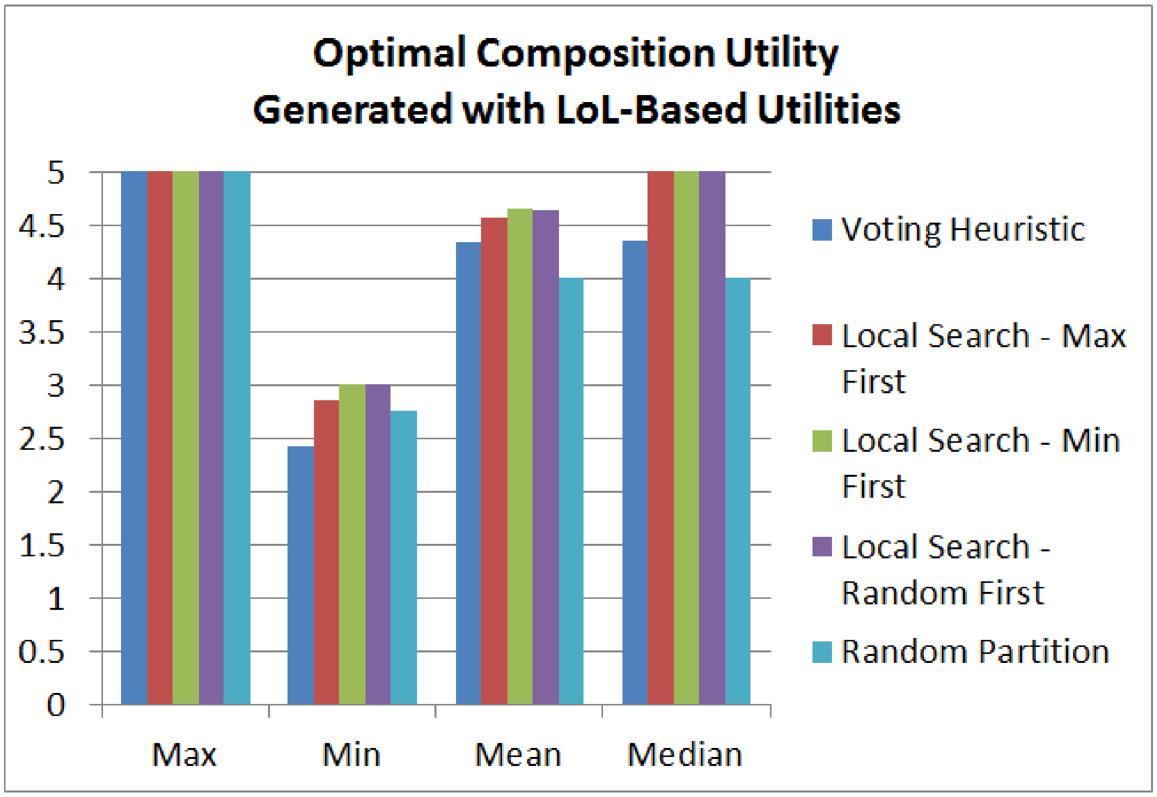

Figure 5.25: Optimal composition utility with generated League of Legends match data using utility function $u_p(r, c) \to (1, 0)$

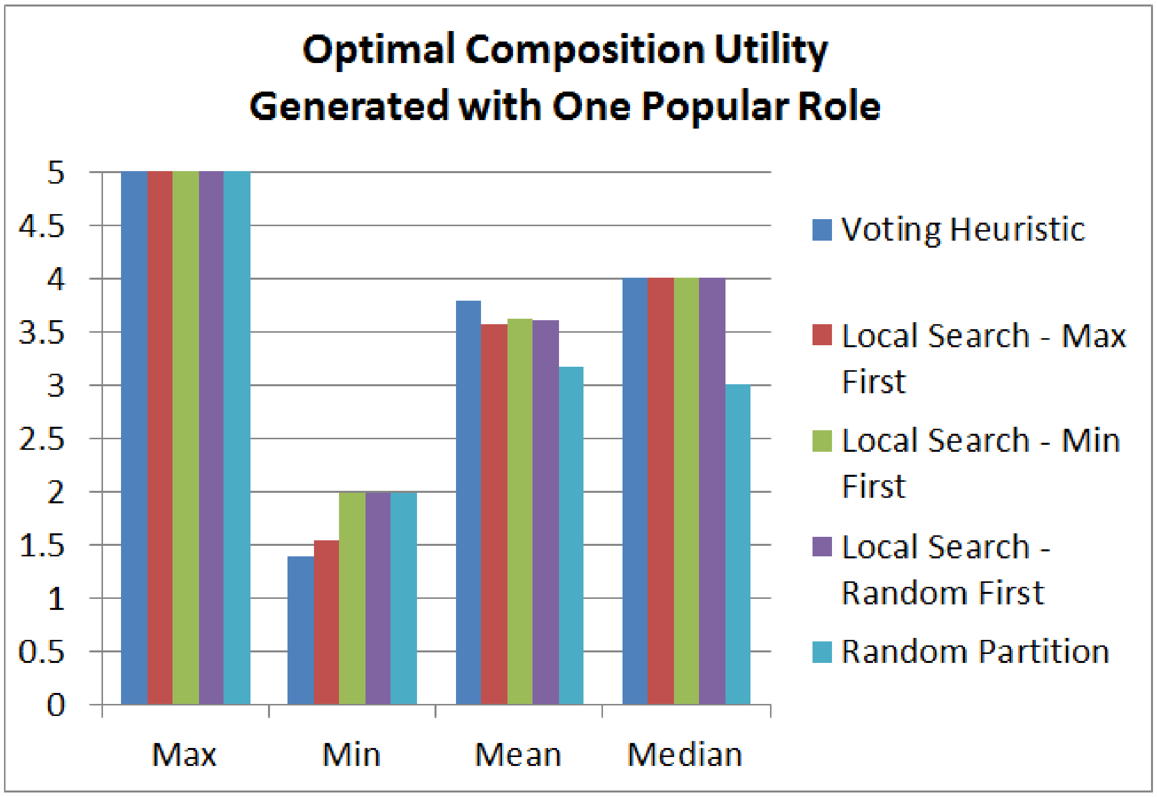

Figure 5.26: Optimal composition utility with generated *one popular role* match data using utility function $u_p(r, c) \rightarrow (1, 0)$

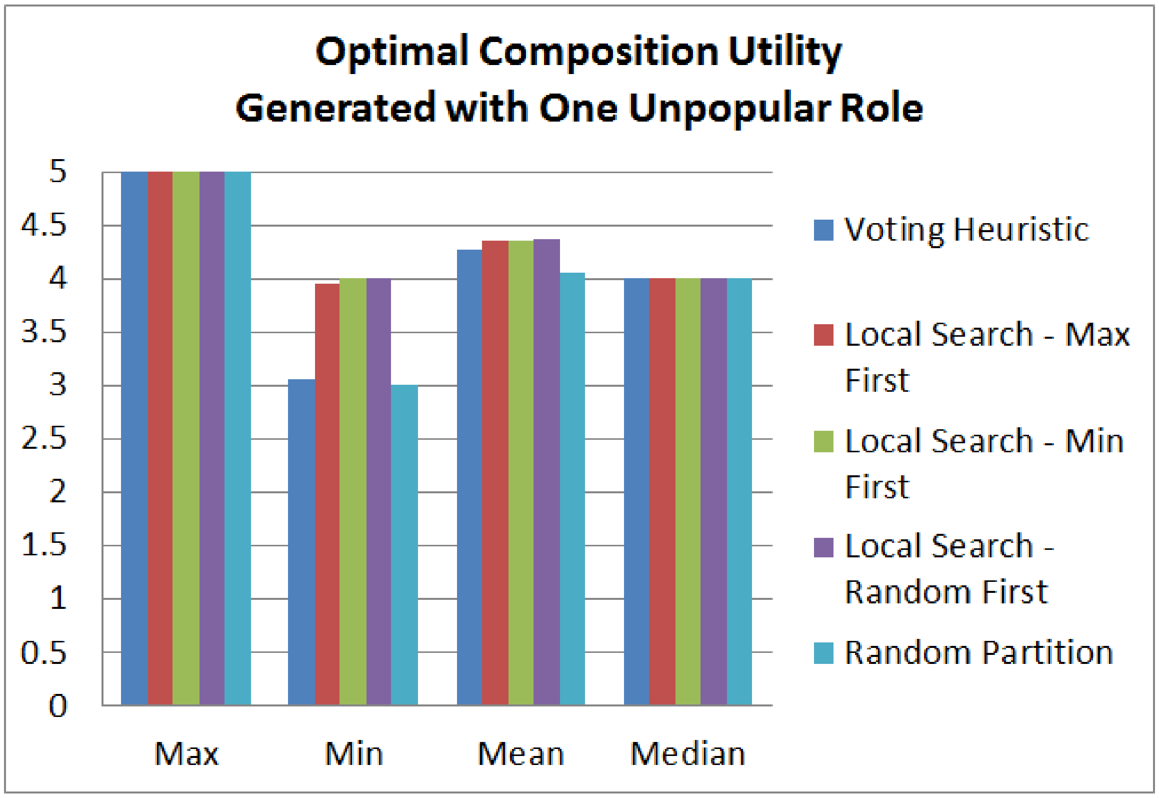

Figure 5.27: Optimal composition utility with generated *one unpopular role* match data using utility function $u_p(r,c) \rightarrow (1,0)$

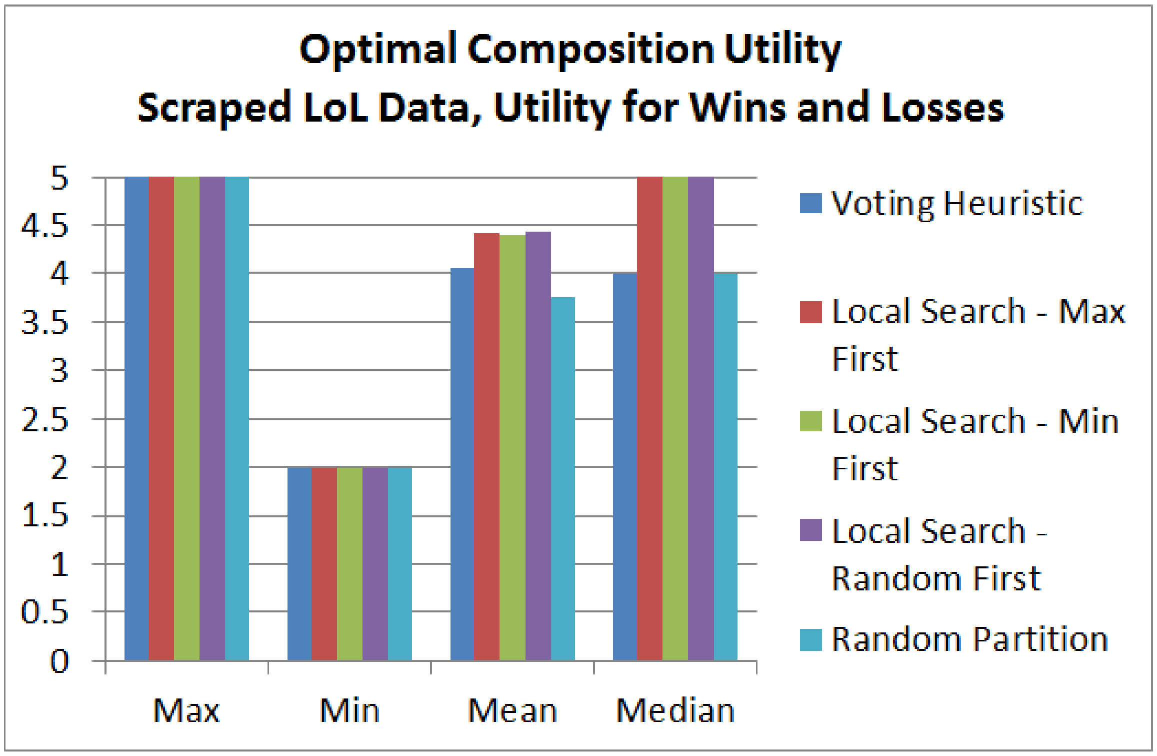

Figure 5.28: Optimal composition utility with League of Legends match data using utility function $u_p(r, c) \rightarrow (1, 0, -1)$

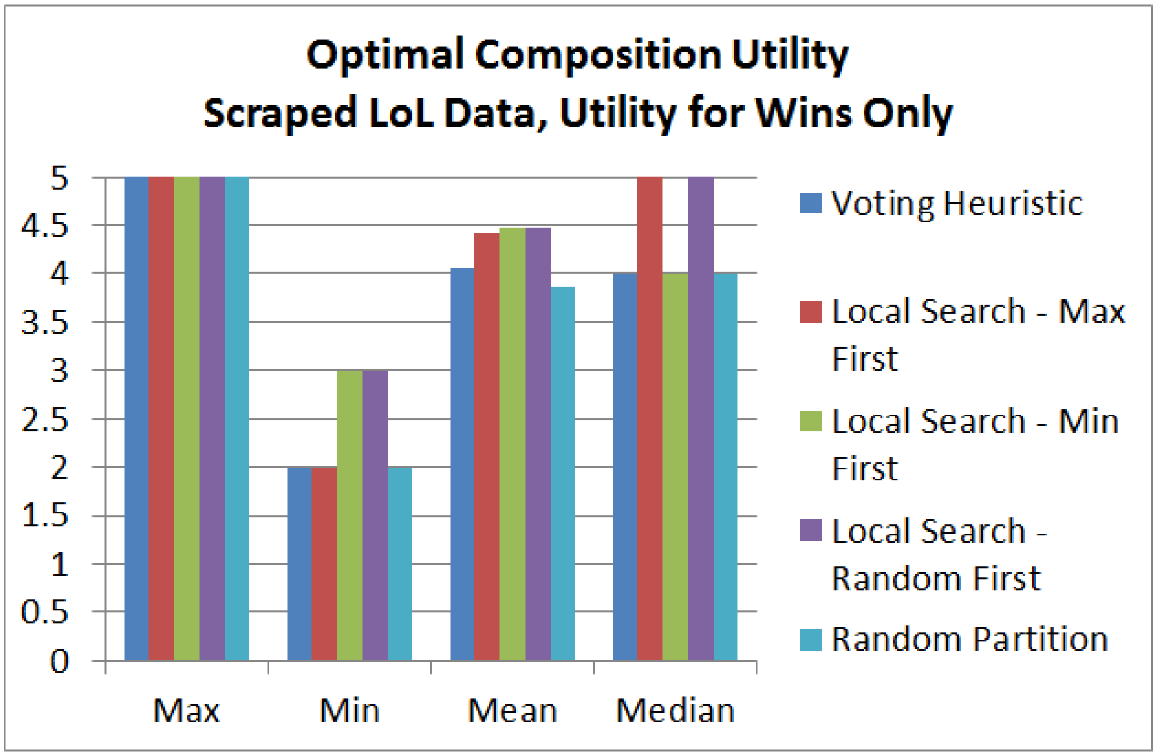

Figure 5.29: Optimal composition utility with League of Legends match data using utility function $u_p(r, c) \rightarrow (1, 0)$

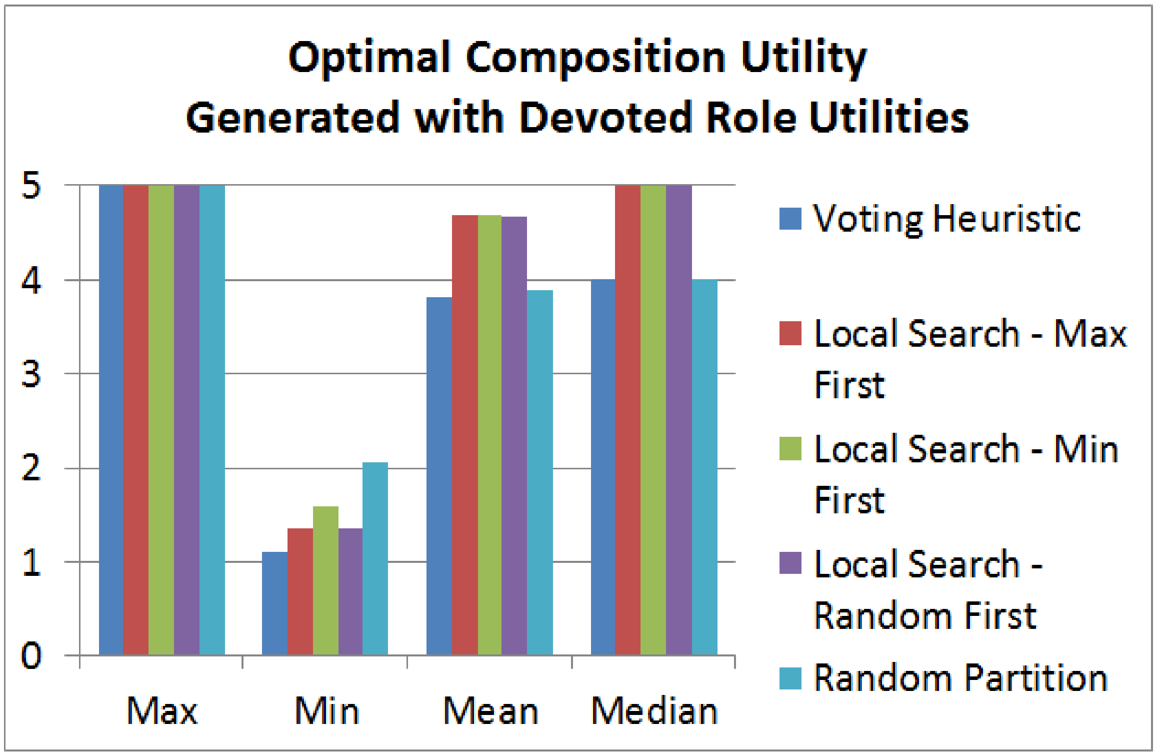

Figure 5.30: Optimal composition utility with generated *devoted role* match data using utility function $u_p(r, c) \rightarrow (1, 0)$

Chapter 6 Conclusions

6.1 Complexity Results

Given a partition, an optimal matching in terms of optimal composition utility can be found in polynomial time as shown in Section 2.6. Similarly, expected utility and expected stability can be evaluated in time polynomial in the RBHG instance; if the number of compositions is polynomial in the number of players and roles, then these evaluations will also be polynomial in the number of players and roles. Each composition can be optimized in cubic time over the team size by the Kuhn Munkres algorithm for square matrices.

Finding a partition and subsequent matching which is *perfect* or optimizes *max sum* or *max min* utility is computationally hard as shown in Section 4.1. I observe that these goal solutions may not be as valuable in a setting where agents can choose to reject a solution. Finding a partition and matching which is stable under several measures of stability is also computationally hard as shown in Section 4.2.

6.2 Algorithms and Utility Distributions

When considering the set of *reachable* and *advocated compositions*, experiments in Section 5.5 show that optimizing expected stability may be a difficult goal to achieve on League of Legends data. It appears that the availability of partitions with a high expected stability is limited given the two utility functions proposed in this paper. The results for optimizing expected utility and optimal composition utility are more promising. By contrast, I discovered stronger results for optimizing *expected stability* over the set of *reachable compositions*. This is particularly pronounced on instances where each agent exhibits a strong devotion to a particular role, and tends to hold up on instances generated from League of Legends matchmaking data and instances generated with utility distributions based upon this data.

My earliest work on optimization showed good results for the greedy voting heuristic [69]. These results hold up somewhat for optimization of optimal composition utility, for which the algorithm was originally designed. However, the new algorithm presented in Section 2.6 for optimizing a matching given a partition allows a random partitioning algorithm to work almost as well as the greedy voting heuristic, as shown in Section 5.5.

The greedy local search heuristic out-performs the greedy voting heuristic for all optimization problems considered in this paper in almost all cases. The greedy local search heuristic also out-performs random partitioning in almost every case. As such, the greedy local search heuristic appears to be the best algorithm so far for selecting a partition.

The greedy local search heuristic can be parallelized both over the set of compositions and the set of agents. The step of checking for which agent to next add to a team can be parallelized by considering agents in parallel. Checking the utility of the optimal matching for each composition can also be parallelized. The results of the algorithm would not be changed, given that the agent utilities for roles and compositions are all independent variables.

6.3 Thoughts on a Utility Functions and a Recommender System

Future work will involve two stages of user testing. First, I will validate the selection of a utility function which represents true utilities of players for roles and compositions. Second, I will design, implement and validate recommendation systems on real users in such role based hedonic game settings as League of Legends.

With a recommendation system, a central authority may recommend a subset of products, services or objects which agents are most likely to accept out of the larger total set of options. Recommendation systems are popular tools in e–commerce [63], social networks [77], medicine [42], and other fields. Applications include movie and video recommendations [26, 54], music recommendations [64, 83], and team recommendation systems [22].

For the particular problem of team recommendation in League of Legends and other role based online video games, I consider the task of recommending roles to players on a team after a partition has been formed. After leveraging the *greedy local search heuristic* to form a partition, the system could recommend optimal role assignments to agents within their teams in polynomial time. Should the agents accept these recommendations, the experiments suggest high max, mean, and median optimal composition utilities can be achieved.

The challenges include making recommendations which adapt to changing preferences and avoiding recommendations to users who either reject the system or are too unpredictable. A successful recommendation system will improve the acceptance rate for optimal composition matchings while not otherwise creating a negative experience. A successful system should also protect itself from possible "trolling" or rejection of the system itself, where a user might consistently act against the recommendations provided. Finally, the system should be able to distinguish between a user actively working against the recommendation system and a user who experiences a major shift in preferences.

I propose a two phase recommendation system. Phase one involves pure learning without recommendation, while phase two incorporates recommendations while continuing to observe changes in user preferences.

Users start in phase one, where they are given no recommendations. The system observes behavior and begins determining user preferences for roles and compositions. Once a user's preferences become sufficiently predictable, that user is moved to phase two. In phase two, users are provided recommendations. The system continues to observe behavior both in terms of user preferences and in terms of acceptance of recommendations. Recommendations will be offered privately to users so that other users do not observe whether or not they accept or reject the recommendations. If a user consistently rejects recommendations, I would expect it to be due to a change in preferences or a rejection of the recommendation system. After a certain threshold of rejections has been reached, a user will be moved back to phase one for additional observation without recommendation. A user who must be moved back to phase one frequently may be kept in phase one for longer periods of time. A user who consistently accepts recommendations in phase two would, alternatively, be kept in phase two.

Users whose preferences experience a significant paradigm shift may move to phase one for a period of time until the recommendation system learns their new preferences, then back to phase two once they are well understood. This would allow the recommendation system to distinguish between changing preferences and behavior which is either highly unpredictable or designed to work against the recommendation system. In this way, users would "opt in" to the recommendation system by generally accepting what it recommends, and "opt out" by ignoring it or actively working against it. Users whose preferences are highly unpredictable are not a good fit for recommendations from the system and would be naturally kept in phase one until such time that their preferences become predictable.

Bibliography

[1] David J. Abraham, Pter Bir, and David F. Manlove. Almost stable matchings in the roommates problem. In Thomas Erlebach and Giuseppe Persinao, editors, *Approximation and Online Algorithms*, volume 3879 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin Heidelberg, 2006.

[2] Hossein Ahmadzadeh and Ellips Masehian. Modular robotic systems: Methods and algorithms for abstraction, planning, control, and synchronization. *Artificial Intelligence*, 223:27–64, 2015.

[3] George A. Akerlof. Social distance and social decisions. *Econometrica: Journal of the Econometric Society*, pages 1005–1027, 1997.

[4] Antonio Albano, Roberto Bergamini, Giorgio Ghelli, and Renzo Orsini. An object data model with roles. In *Very Large Data Bases*, volume 93, pages 39–51. Citeseer, 1993.

[5] Craig A. Anderson and Brad J. Bushman. Effects of violent video games on aggressive behavior, aggressive cognition, aggressive affect, physiological arousal, and prosocial behavior: A meta-analytic review of the scientific literature. *Psychological Science*, 12(5):353–359, 2001.

[6] Haris Aziz and Florian Brandl. Existence of stability in hedonic coalition formation games. In *Proc. Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012),*, 2012.

[7] Haris Aziz, Felix Brandt, and Paul Harrenstein. Fractional hedonic games. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, pages 5–12. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[8] Haris Aziz, Felix Brandt, and Hans Georg Seedig. Optimal partitions in additively separable hedonic games. *arXiv preprint arXiv:1005.4540*, 2010.

[9] Haris Aziz, Felix Brandt, and Hans Georg Seedig. Stable partitions in additively separable hedonic games. In *Proc. AAMAS '11*, pages 183–190, 2011.

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

[11] José Baca, S. G. M. Hossain, Prithviraj Dasgupta, Carl A. Nelson, and Ayan Dutta. Modred: Hardware design and reconfiguration planning for a high dexterity modular self-reconfigurable robot for extra-terrestrial exploration. *Robotics and Autonomous Systems*, 62(7):1002–1015, 2014.

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

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

[14] Vittorio Bilò, Angelo Fanelli, Michele Flammini, Gianpiero Monaco, and Luca Moscardelli. On the price of stability of fractional hedonic games. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1239–1247. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[15] Péter Biró, Elena Inarra, and Elena Molis. A new solution concept for the roommate problem: Q-stable matchings. *Under review*, 2015.

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

[17] Christian Borgelt. Keeping things simple: Finding frequent item sets by recursive elimination. In *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, pages 66–70. ACM, 2005.

[18] Simina Brânzei and Kate Larson. Social distance games. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1281–1282. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[19] Arthur P. Brief and Stephan J. Motowidlo. Prosocial organizational behaviors. *Academy of Management Review*, 11(4):710–725, 1986.

[20] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.

[21] ZAM Network L. L. C. League of legends summoner stats & champion build guides - lolking. ZAM Network L. L. C., 2015. `http://www.lolking.net/`, Date accessed: 2015.

[22] Yen-Liang Chen, Li-Chen Cheng, and Ching-Nan Chuang. A group recommendation system with consideration of interactions among group members. *Expert Systems with Applications*, 34(3):2082–2090, 2008.

[23] Bryan K. Church, R. Lynn Hannan, and Xi Jason Kuang. Shared interest and honesty in budget reporting. *Accounting, Organizations and Society*, 37(3):155–167, 2012.

[24] Vincent Conitzer and Tuomas Sandholm. Communication complexity of common voting rules. In *Proceedings of the 6th ACM Conference on Electronic Commerce*, EC '05, pages 78–87, New York, NY, USA, 2005. ACM.

[25] Andreas Darmann, Edith Elkind, Sascha Kurz, Jérôme Lang, Joachim Schauer, and Gerhard Woeginger. Group activity selection problem. In *Internet and Network Economics*, pages 156–169. Springer, 2012.

[26] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 293–296. ACM, 2010.

[27] Effrosyni Diamantoudi and Licun Xue. Farsighted stability in hedonic games. *Social Choice and Welfare*, 21(1):39–61, 2003.

[28] Dinko Dimitrov, Peter Borm, Ruud Hendrickx, and Shao Chin Sung. Simple priorities and core stability in hedonic games. *Social Choice and Welfare*, 26(2):421–433, 2006.

[29] J. H. Drèze and J. Greenberg. Hedonic coalitions: optimality and stability. *Econometrica*, 48(4):987–1003, 1980.

[30] Nick Eckenstein and Mark Yim. Modular reconfigurable robotic systems: lattice automata. In *Robots and Lattice Automata*, pages 47–75. Springer, 2015.

[31] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[32] Blizzard Entertainment. Diablo iii official game site - battle.net. Blizzard Entertainment, 2015. `http://us.battle.net/d3/en/`, Date accessed: 2015.

[33] Blizzard Entertainment. Heroes of the storm official game site. Blizzard Entertainment, 2015. `http://us.battle.net/heroes/en/`, Date accessed: 2015.

[34] Blizzard Entertainment. World of warcraft – battle.net. Blizzard Entertainment, 2015. `http://us.battle.net/wow/en/`, Date accessed: 2015.

[35] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(5):2015–2028, 2004.

[36] Martin Gairing and Rahul Savani. Computing stable outcomes in hedonic games. In *Algorithmic Game Theory*, pages 174–185. Springer, 2010.

[37] Oded Goldreich. *Computational Complexity, a Conceptual Perspective.* Cambridge University Press, 2008.

[38] Georg Gottlob, Michael Schrefl, and Brigitte Röck. Extending object-oriented systems with roles. *ACM Transactions on Information Systems (TOIS)*, 14(3):268–296, 1996.

[39] Marco Guazzone, Cosimo Anglano, and Matteo Sereno. A game-theoretic approach to coalition formation in green cloud federations. In *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2014*, pages 618–625. IEEE, 2014.

[40] Tyler Gunn and John Anderson. Dynamic heterogeneous team formation for robotic urban search and rescue. *Journal of Computer and System Sciences*, 81(3):553–567, 2015.

[41] Martin Hoefer, Daniel Váz, and Lisa Wagner. Hedonic coalition formation in networks. In *Twenty-Ninth Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, 2015.

[42] T Ryan Hoens, Marina Blanton, Aaron Steele, and Nitesh V Chawla. Reliable medical recommendation systems with patient privacy. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(4):67, 2013.

[43] S. G. M. Hossain, Carl A. Nelson, Khoa D. Chu, and Prithviraj Dasgupta. Kinematics and interfacing of modred: A self-healing capable, four-dof modular self-reconfigurable robot. *JMR*, 13:1256, 2014.

[44] Riot Games Inc. League of legends. Riot Games Inc., 2015. `http://na.leagueoflegends.com/`, Date accessed: 2015.

[45] Hi-Rez Studios Incorporated. Smite landing page. Hi-Rez Studios Incorporated, 2015. `https://account.hirezstudios.com/smitegame/default.aspx?ok=`, Date accessed: 2015.

[46] Riot Games Incorporated. League of Legends' growth spells bad news for Teemo — Riot Games. Riot Games Incorporated, 2012. `http://www.riotgames.com/articles/20121015/138/league-legends-growth-spells-bad-news-teemo`, Date accessed: 2015.

[47] Robert W. Irving. An efficient algorithm for the stable roommates problem. *Journal of Algorithms*, 6(4):577–595, 1985.

[48] Robert W. Irving and David F. Manlove. The stable roommates problem with ties. *Journal of Algorithms*, 43(1):85–105, 2002.

[49] Sungwook Kim. An adaptive smart grid management scheme based on the coopetition game model. *Electronics and Telecommunications Research Institute Journal*, 36(1):80–88, 2014.

[50] Jérôme Lang, Anja Rey, Jörg Rothe, Hilmar Schadrack, and Lena Schend. Representing and solving hedonic games with ordinal preferences and thresholds. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1229–1237. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[51] Hooyeon Lee and Yoav Shoham. Stable invitations. *Twenty-Ninth Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, 2015.

[52] Ido Liviatan, Yaacov Trope, and Nira Liberman. Interpersonal similarity as a social distance dimension: Implications for perception of others' actions. *Journal of Experimental Social Psychology*, 44(5):1256–1269, 2008.

[53] J. Marden and Adam Wierman. Overcoming the limitations of utility design for multiagent systems. *IEEE Transactions on Automatic Control*, 58(6):1402–1415, 2013.

[54] Nicholas Mattei, James Forshee, and Judy Goldsmith. An empirical study of voting rules and manipulation with large datasets. *Computational Social Choice*, 2012.

[55] Charles A. O'Reilly and Jennifer Chatman. Organizational commitment and psychological attachment: The effects of compliance, identification, and internalization on prosocial behavior. *Journal of Applied Psychology*, 71(3):492, 1986.

[56] Dominik Peters and Edith Elkind. Simple causes of complexity in hedonic games. Proceedings of the Twenty-Forth International Joint Conference on Artificial Intelligence, 2015.

[57] Boris G. Pittel and Robert W. Irving. An upper bound for the solvability probability of a random stable roommates instance. *Random Structures & Algorithms*, 5(3):465–486, 1994.

[58] Raptr. The official raptr blog home — most played PC games: February 2015 heroes of the storm makes its debut; dragon age: Inquisition plummets. Raptr, 2015. `http://caas.raptr.com/most-played-pc-games-february-2015-heroes-of-the-storm-makes-its-debut-dragon-age-inquisition-plummets/`, Date accessed: 2015.

[59] Walid Saad, Zhu Han, Tamer Basar, M. Debbah, and Are Hjorungnes. Hedonic coalition formation for distributed task allocation among wireless agents. In *Proc. IEEE Transactions on Mobile Computing*, 2011.

[60] Walid Saad, Zhu Han, Tamer Basar, Mérouane Debbah, and Are Hjorungnes. Hedonic coalition formation for distributed task allocation among wireless agents. *IEEE Transactions on Mobile Computing*, 10(9):1327–1344, 2011.

[61] Walid Saad, Zhu Han, Tamer Basar, Are Hjorungnes, and Ju Bin Song. Hedonic coalition formation games for secondary base station cooperation in cognitive radio networks. In *Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2010.

[62] Walid Saad, Zhu Han, Are Hjørungnes, Dusit Niyato, and Ekram Hossain. Coalition formation games for distributed cooperation among roadside units in vehicular networks. *IEEE Journal on Selected Areas in Communications, Special issue on Vehicular Communications and Networks*, 2011.

[63] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 158–167. ACM, 2000.

[64] Il-hyung Shin, Jaepyeong Cha, Gyeong Woo Cheon, Choonghee Lee, Seung Yup Lee, Hyung-Jin Yoon, and Hee Chan Kim. Automatic stress-relieving music recommendation system based on photoplethysmography-derived heart rate variability analysis. In *Engineering in Medicine and Biology Society, 2014 36th Annual International Conference of the IEEE*, pages 6402–6405. IEEE, 2014.

[65] Valve Software. Counter–strike: Global offensive. Valve Software, 2015. `http://blog.counter-strike.net/`, Date accessed: 2015.

[66] Valve Software. Dota 2. Valve Software, 2015. `http://blog.dota2.com/`, Date accessed: 2015.

[67] Haeyeop Song and Jaemin Jung. Antecedents and consequences of gender swapping in online games. *Journal of Computer-Mediated Communication*, 20(1):434–449.

[68] Matthew Spradling and Judy Goldsmith. Stability in role based hedonic games. In *The Twenty-Eighth International Florida Artificial Intelligence Research Society Conference*. Springer, 2015.

[69] Matthew Spradling, Judy Goldsmith, Xudong Liu, Chandrima Dadi, and Zhiyu Li. Roles and teams hedonic game. In *Proc. Algorithmic Decision Theory*, pages 351–362. Springer, 2013.

[70] Keith Starcher and Dennis Proffitt. Encouraging students to read: What professors are (and aren't) doing about it. *International Journal of Teaching and Learning in Higher Education*, 23(3):396–407, 2011.

[71] Frostburn Studios. Heroes of newerth – home. Frostburn Studios, 2015. `http://www.heroesofnewerth.com/`, Date accessed: 2015.

[72] Shao-Chin Sung and Dinko Dimitrov. On core membership testing for hedonic coalition formation games. *Operations Research Letters*, 35(2):155–158, 2007.

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

[74] Long Tran-Thanh, Tri-Dung Nguyen, Talal Rahwan, Alex Rogers, and Nicholas R. Jennings. An efficient vector-based representation for coalitional games. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 383–389. AAAI Press, 2013.

[75] Elio Tuci, Roderich Groß, Vito Trianni, Francesco Mondada, Michael Bonani, and Marco Dorigo. Cooperation through self-assembly in multi-robot systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(2):115–150, 2006.

[76] Norovsambuu Tumennasan. Moral hazard and stability. *Social Choice and Welfare*, 43(3):659–682, 2014.

[77] Frank Edward Walter, Stefano Battiston, and Frank Schweitzer. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 16(1):57–74, 2008.

[78] Chih-Chien Wang and Chia-Hsin Wang. Helping others in online games: Prosocial behavior in cyberspace. *CyberPsychology & Behavior*, 11(3):344–346, 2008.

[79] C. Wei. *Cognitive coordination for cooperative multi-robot teamwork*. PhD thesis, TU Delft, Delft University of Technology, 2015.

[80] Mason Wright and Yevgeniy Vorobeychik. Mechanism design for team formation. *Twenty-Ninth Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, 2015.

[81] K. Xu, K. Wang, R. Amin, J. Martin, and R. Izard. A fast cloud-based network selection scheme using coalition formation games in vehicular networks. *IEEE Transactions on Vehicular Technology*, 2014.

[82] Mark Yim, Kimon Roufas, David Duff, Ying Zhang, Craig Eldershaw, and Sam Homans. Modular reconfigurable robots in space applications. *Autonomous Robots*, 14(2-3):225–237, 2003.

[83] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. Auralist: introducing serendipity into music recommendation. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mning*, pages 13–22. ACM, 2012.

[84] Michael Zuckerman, Ariel D. Procaccia, and Jeffrey S. Rosenschein. Algorithms for the coalitional manipulation problem. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 277–286. Society for Industrial and Applied Mathematics, 2008.

Vita

Born Matthew Jordan Spradling in Mount Sterling, Kentucky.

Education

Ph. D. Computer Science (Pursuant)	University of Kentucky	August 2015
B. S. Computer Science	University of Kentucky	May 2010
B. S. Business Administration Minor in Philosophy	University of Louisville	Dec 2005

Professional Positions

University of Kentucky Department of Computer Science

- Research Assistant, 2013-2015
- Teaching Assistant, 2009-2014

Scholastic and Professional Honors

- Verizon fellowship, University of Kentucky Department of Computer Science, 2015.
- *Nominated*, University of Kentucky Association for Computing Machinery Outstanding Teaching Assistant Award, 2014.
- Graduate Certificate in College Teaching and Learning, University of Kentucky Graduate School, 2014.
- Duncan E. Clarke Memorial Innovation Award, University of Kentucky Department of Computer Science, 2013.

Professional Publications

- Spradling, M., Goldsmith, J., *Stability in Role Based Hedonic Games*, 28th International Florida Artificial Intelligence Research Society Conference, 2015.
- Spradling, M., *Roles and Teams Hedonic Games*, 19th AAAI Doctoral Consortium, 2014.
- Spradling, M., Goldsmith, J., Liu, X., Dadi, C., & Li, Z. *Roles and Teams Hedonic Game*, 3rd International Conference on Algorithmic Decision Theory, 2013.
  Also presented without proceedings at 7th Multidisciplinary Workshop on Advances in Preference Handling at IJCAI'13, 2013.