Optimizing Expected Utility and Stability in Role Based Hedonic Games

Matthew Spradling
University of Michigan - Flint
mjspra@umflint.edu

Abstract

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 roles she and her teammates fulfill within the coalition. I show positive results for finding optimal or stable role matchings given a partitioning into teams. In settings such as massively multiplayer online games, a central authority assigns agents to teams but not necessarily to roles within them. For such settings, I consider the problems of optimizing *expected utility* and *expected stability* in RBHG. I show that the related optimization problems for partitioning are NP-hard. I introduce a local search heuristic method for approximating such solutions. I validate the heuristic by comparison to existing partitioning approaches using real-world data scraped from League of Legends games.

Introduction

In coalition formation games, agents from a population have various utilities for different partitions, where a partition is a set of teams (subsets of agents). Hedonic games are coalition formation games in which an agent's preference for a partition is determined only by the coalition to which the agent is assigned (Sung and Dimitrov 2010). Hedonic games are used to model matching of agents to teams when agents only care about the utility of their own teams, not others.

Stability of an assignment is of primary importance when agents are able to leave the team if they are not satisfied (Spradling and Goldsmith 2015). While a utility-function maximizing partition is attractive, forming such a partition is not helpful if the agents cannot be guaranteed to keep the assignment. An optimal assignment may not be stable. It has been observed that, for some stability measures, there may be multiple stable assignments with variable utility levels (Spradling et al. 2013). A stable assignment is not guaranteed to have an optimal utility among all stable assignments.

In this paper, I consider the variant *Role Based Hedonic Game* (RBHGs) (Spradling and Goldsmith 2015). In this model, an agent's utility for a partitions 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*.

Table 1: Ex. RBHG instance with $|P| = 4, R = \{A, B\}$

$\langle r, c\rangle$	$u_{p_0}(r, c)$	$u_{p_1}(r, c)$	$u_{p_2}(r, c)$	$u_{p_3}(r, c)$
$\langle A, AA\rangle$	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

As per (Spradling and Goldsmith 2015), a *Role Based Hedonic Game* (RBHG) instance consists of:

- P: A population of players
- R: A set of roles
- C: A set of compositions, where a composition $c \in C$ is a multiset (bag) of roles from R.
- $U : P \times R \times C \to \mathbb{Z}$ defines the utility function $u_i(r, c)$ for each player p_i. Generally, I assume that for all $p_i \in P$ and for all $r \in R$, $u_i(r, \{r\}) = 0$.

A solution to an RBHG instance is a partition π of each agent into a team (set of agents) t and a matching M of agents to roles in R. Because RBHG is a hedonic game, utility of a player for a partition π and matching M is equal to that player's utility for $(\pi(p_i), M(p_i)) = (r^i, c^i, t^i)$. I will refer to a player p_i's utility for (π, M) as $u_i(r^i, c^i)$.

This work is motivated by team formation among autonomous drones (Hock and Schoellig 2016) and by human team formation in massively multiplayer online games (MMOs). A drone may be specially suited to fulfilling a subset of roles needed by a local swarm. It must efficiently determine which group to assist and in what capacity as new tasks arise. In the paradigm of MMOs, the top five online games in the world each incorporate some variation of role based team formation. Including *Overwatch* and *League of Legends*, these games accounted for 259.22 million hours of combined viewing over the Twitch streaming service worldwide in January, 2017 alone (Gamoloco 2017). In both settings, role assignments are chosen by the agents after teams or coordinates are determined by a central authority. An agent then chooses a role depending upon an internal utility function. For this reason, I focus on the problems of finding partitions which have *optimal expected stability* and *optimal expected utility* over the possible matchings.

I provide polynomial time algorithms for finding optimal and stable matchings M given a partitioning π of agents into teams. I consider the complexity of recognizing partitions with *optimal expected stability* and *optimal expected utility*, both of which I prove to be NP-complete for *MaxSum* and *MaxMin* optimization. I introduce a *greedy local search* heuristic which runs in polynomial time. I validate the heuristic by comparison to *"what if analysis"* and a previously described *greedy voting heuristic* (Spradling et al. 2013) using real world matchmaking data from League of Legends games. The experiments show improvements to expected utility, expected stability, and to the percentage of teams with a stable matching. In particular, greedy local search identifies *stable role and team recommendations for over* 50% *of players in the test population*. These results indicate that greedy local search can be applied to real world RBHG matchmaking scenarios.

Related Work: Stability and Optimization in Coalition Formation Games

In economics, hedonic coalition formation games model situations where agents join teams to collaboratively produce goods for themselves (Drèze and Greenberg 1980). Hedonic game models have been applied wide ranging fields, including distributed task allocation in wireless agents (Saad et al. 2011a), communications networks (Saad et al. 2010), vehicular networks (Saad et al. 2011b; Xu et al. 2015), adaptive smart grid management (Kim 2014), federation formation among cloud providers (Guazzone, Anglano, and Sereno 2014), and measuring impact on perceived learning outcomes (Alexiou and Doerga 2015).

For general hedonic games, finding a core stable coalition structure is NP-complete (Ballester 2004). Farsighted stability in general hedonic games considers a case where agents avoid defecting from a team if the defection would induce a worse change by agents on their new or former team (Diamantoudi and Xue 2003). It has been shown that core stable and Nash stable solutions are farsighted stable while individually stable and contractually individually stable solutions do not guarantee this (Diamantoudi and Xue 2003).

Work in role based hedonic games first considered the *roles and teams hedonic game* (RTHG) model, a special case of RBHG in which all compositions are of equal size. Even under this constraint, finding a perfect, maxsum, or maxmin optimal partition is NP-hard (Spradling et al. 2013). While optimization would be valuable, agents in online games may choose not to play and reject an unstable partition.

For general RBHG several stable partitioning problems are NP-complete or coNP-complete (Spradling and Goldsmith 2015). Similar results have been found for stability and optimization problems related to additively separable hedonic games (Sung and Dimitrov 2010; Aziz, Brandt, and Seedig 2010; 2011), hedonic games represented by an individually rational list of coalitions (Ballester 2004), anonymous hedonic games (Banerjee, Konishi, and Sönmez 2001) the Group Activity Selection Problem (GASP) (Darmann et al. 2012), and hedonic games with dichotomous preferences among others (Peters 2016). Altruistic hedonic games provide a model in which stable partitions of several forms can be found when the agents seek to help or avoid harming allies in the network (Nguyen et al. 2016).

Past work considers a scenario where all partitioning is performed by the central authority. In massively multiplayer online games, agents may be partitioned to teams by the server but not necessarily the roles within them. Once partitioned, the matching is determined by the agents while the game is in progress. Even if agents are assigned to a partition which contains an optimal *and* stable matching, it is not guaranteed that the agents will stabilize on that particular matching. The match may end, or someone may quit playing, before agents discover how they can collaborate.

For human agents, I consider the problem of finding partitions in which players are *most likely* to stabilize upon a matching. I call this the *optimal expected stability partition*. Similarly, I consider the problem of finding the *optimal expected utility partition* in which the *average* optimal composition utility for each team is maximized.

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 each composition.

Definition 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 matching of agents to roles within c which maximizes the sum of the utilities for all agents on that team.*

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.*

Theorem 1 *When agents are matched into roles by a central authority, an optimal or stable composition for a team t can be identified in polynomial time.*

Proof 1 *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 (Edmonds and Karp 1972).*

The optimal composition ${c_t}^o \in C$ *of t can be identified in time $\mathcal{O}(|C| \cdot |t|^3)$ by computing optimal utility for each $c \in C$. When agents have 1-0 "accept/-reject" utilities, a* stable composition *will have optimal utility 1 for each $p_i \in t$.* □

Without a central authority to assign agents to roles within a team t, agents in t will either stabilize upon 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 compositions.

Definition 2 *An* acceptable composition ${c_t}^a$ *for a team t is one for which $u_p(r, {c_t}^a) > 0$ for at least one agent in $p \in t$.*

For a 0-1 instance of RBHG where agents either accept *or* reject *each role and composition pair, a* stable composition ${c_t}^s$ *for a team t is one for which $u_p(r, {c_t}^s) = 1$ for each agent $p \in t$ given an optimal utility matching of roles.*

Observe that any stable composition *is also an* acceptable composition *but that the reverse is not necessarily the case.*

Definition 3 *Let ${q_t}^s$ be the quantity of stable compositions for t and ${q_t}^a$ be the quantity of acceptable compositions for t. The* expected stability *of t is 0 if ${q_t}^a = 0$ or the ratio ${q_t}^s/{q_t}^a$ for ${q_t}^a > 0$.*

Let ${u_t}^a$ be the sum of optimal utilities for all ${q_t}^a$ acceptable compositions for t. The expected utility *of t is 0 if ${q_t}^a = 0$ or the ratio ${u_t}^a/{q_t}^a$ for ${q_t}^a > 0$.*

The goal it this setting is to select a partition π which optimize *expected utility*, *expected stability*, or the percent of teams for which there is a *stable composition*.

Complexity of Optimization

In this section, I define the decision problems related to finding optimal expected utility and optimal expected stability partitions in RBHG. I prove each to be NP-complete by reduction from PERFECT RBHG.

Definition 4 *An instance of Binary RBHG is an instance of RBHG such that for each agent $p \in P$, for each $c \in C$, for each $r \in c$, $u_p(r, c) \rightarrow \{0, 1\}$.*

A perfect *solution (π, M) of an instance of Binary RBHG has the property that, for each $p \in P$, $u_p(r^p, c^p) = 1$.*

Definition 5 *The language* PERFECT RBHG *consists of those instances of RHBG for which a perfect solution exists.* PERFECT BINARY RBHG *consists of those instances of Binary RBHG for which a perfect solution exists.*

Observation 1 *An instance $A = \langle P, R, C, U, m\rangle$ of RTHG is an instance $B = f(A) = \langle P, R, C, U\rangle$ of RBHG where the set C of compositions consists of all multisets of m roles within R. Therefore, an instance A of RTHG has a perfect solution iff $B = f(A)$ has the same perfect solution. Therefore* PERFECT RTHG $\leq_m^P$ PERFECT RBHG.

Observation 2 PERFECT RTHG *is NP-complete even for special cases where utilities are binary (Spradling et al. 2013). Therefore* PERFECT RBHG *and* PERFECT BINARY RBHG *are both NP-hard.*

Definition 6 *The language* MAXMIN ES RBHG *consists of pairs (B, k), where B is an instance of RBHG, k is an integer, and there exists a partition π of B such that for each $t \in \pi$ expected stability is $\geq k$.*

Theorem 2 MAXMIN ES RBHG *is NP-complete.*

Proof 2 *To show that* MAXMIN ES RBHG *is in NP, consider the following NP algorithm. Given an instance of* MAXMIN ES RBHG*, guess a partition π and evaluate the average expected stability value. This can be computed in time $\mathcal{O}(|P| \cdot |C| \cdot \text{MAX}(|t|)^2)$. Observe that $1 \leq |P|$ and that there are at most $|P|$ teams. Observe that $|C|$ is polynomial in the size of the instance B of RBHG. For each team $t \in \pi$, for each composition $c \in C$, compute the optimal expected stability value of t for each $p \in t$ given c. Each of the $(|P| \cdot |C|)/t$ square matrices can be optimized in time $\mathcal{O}(|t|^3)$. Stop and reject if the sum total expected stability value of all teams $< k$, otherwise accept. This checking is in time polynomial in the size of the input.*

To show NP-hardness, I show that PERFECT BINARY RBHG $\leq_m^P$ MAXSUM ES RBHG. *It other words, given an instance $A = \langle P, R, C, U\rangle$ of* PERFECT BINARY RBHG*, I construct an instance $f(A) = \langle P', R', C', U', k\rangle$ of* MAXMIN ES RBHG *such that $A \in$* PERFECT BINARY RBHG *iff $f(A) \in$* MAXMIN ES RBHG.

Let $B = \langle P, R, C, U\rangle$ be an instance of PERFECT BINARY RBHG. *I construct an instance $f(B) = \langle P', R', C', U', k\rangle$ of* MAXMIN ES RBHG *as follows. Let $P' = P$, $R' = R$, $C' = C$, $U' = U$, and let $k = 1/|C|$.*

I claim that there is a π' of $f(B)$ s.t. each $t \in \pi'$ has expected stability $\geq k$ iff there is a (π, M) of B s.t. $B \in$ PERFECT BINARY RBHG.

Suppose there is a solution π' of $f(B)$ such that $f(B) \in$ MAXMIN ES RBHG. *Then I construct a solution (π, M) of B as follows. Let $\pi = \pi'$. For each $t \in \pi'$, for each $c \in C$, find the optimal role assignment m for (c, t) in time $\mathcal{O}(|t|^3)$ and add m to M. Given the assumption that each $t \in \pi'$ has expected stability $\geq 1/|C|$, there must be at least one of the $|C|$ compositions which has a perfect assignment for t. Therefore (π, M) is a perfect solution for B. Thus $f(B) \in$* MAXMIN ES RBHG *implies that $B \in$* PERFECT BINARY RBHG.

Suppose there is a solution (π, M) of B such that $B \in$ PERFECT BINARY RBHG. *Then I construct a solution π of $f(B)$ as follows. Let $\pi' = \pi$. Given the assumption that M is a perfect matching, it must be the case that each $t \in \pi$ has a perfect assignment for as least one of the $|C|$ compositions. Therefore π' is a partition of $f(B)$ s.t. each $t \in \pi$ has expected stability $\geq 1/|C|$. Thus $B \in$* PERFECT BINARY RBHG *implies that $f(B) \in$* MAXMIN ES RBHG.

Therefore MAXMIN ES RBHG *is NP-hard.* □

Definition 7 *The language* MAXSUM ES RBHG *consists of pairs (B, k), where B is an instance of RBHG, k is an integer, and there exists a partition π of B such that the sum total expected stability of all $t \in \pi$ is $\geq k$.*

Theorem 3 MAXSUM EU RBHG *is NP-complete.*

Proof Sketch To show that MAXSUM ES RBHG is in NP, consider the following NP algorithm. Given an instance of MAXSUM ES RBHG, guess a partition π and evaluate the sum total expected stability over all teams in π. This can be computed in time $\mathcal{O}(|P| \cdot |C| \cdot \text{MAX}(|t|)^2)$. Stop and reject if the average expected utility value is $< k$, otherwise accept. This checking is in time polynomial in the size of the input.

To show NP-hardness, I show that PERFECT BINARY RBHG $\leq_m^P$ MAXSUM ES RBHG. The reduction is similar to PERFECT BINARY RBHG $\leq_m^P$ MAXMIN ES RBHG. □

Definition 8 *The language* MAXMIN EU RBHG *consists of pairs (B, k), where B is an instance of RBHG, k is an integer, and there exists a partition π of B such that for each $t \in \pi$ expected utility is $\geq k$.*

The language MAXSUM EU RBHG *consists of pairs (B, k), where B is an instance of RBHG, k is an integer, and there exists a partition π of B such that the sum total expected utility of all $t \in \pi$ is $\geq k$.*

Theorem 4 MAXSUM EU RBHG *and* MAXMIN EU RBHG *are both NP-complete.*

Proof Sketch Both MAXSUM EU RBHG and MAXMIN EU RBHG are polynomial time verifiable by algorithms

similar to those utilized for MAXSUM ES RBHG and MAXMIN ES RBHG. Guess a partition then verify the expected utility of each team, which can be computed in time polynomial in the size of the input.

To show NP-hardness, I show that MAXSUM ES RBHG $\leq_m^P$ MAXSUM EU RBHG and that MAXMIN EU RBHG $\leq_m^P$ MAXMIN ES RBHG. □

Heuristic Matchmaking Methods

In this section I outline three matchmaking methods for RBHG: "What if" analysis, greedy voting (Spradling et al. 2013), and greedy local search.

"What if" Analysis

"What if" analysis generates a partition uniformly at random and computes the value of *expected stability*, *expected utility*, or some other optimization criteria. If there is improvement over the previous best the new partition is kept. This procedure is repeated with a new partition until all $|P|!$ partitions have been enumerated, until some threshold value δ is met for the optimization criteria, or until some maximum number of partitions Δ have been evaluated.

Each iteration of "what if" analysis requires $\mathcal{O}(|P|/m)$ computations of the optimization metric being considered, where m is the maximum team size. Given that computing expected utility or stability for a team has a cost of $\mathcal{O}(|C| \cdot m^3)$, the total cost of considering one partition for this optimization goal is $\mathcal{O}(|P| \cdot |C| \cdot m^2)$. If Δ is left uncapped, this algorithm is guaranteed to return a partition nearest to δ. In the worst case, this requires $\mathcal{O}(|P|!)$ iterations to enumerate all partitions of P.

Greedy Voting

Greedy voting considers agent utilities as *scoring rule* style votes in an election over the available compositions (Conitzer and Sandholm 2005). Agent utilities for each compositions, over the roles within that composition, are added together. The composition $c \in C$ with the highest total score is selected along with the $|c|$ agents that most preferred c. This team is added to the partition and their scores for all compositions are subtracted from the totals. The procedure repeats with a new vote over the compositions and a new team being formed until all players have been matched. The greedy voting heuristic has a running time of $\mathcal{O}(|P|^2/m)$, or $\mathcal{O}(|P| \cdot |C| \cdot m)$ if $|P| < |C| \cdot m^2$ (Spradling et al. 2013).

Greedy Local Search

I compare to the following greedy local search heuristic. Select an agent $p \in P$ as the *pivotal agent* and locally optimize the chosen metric 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 $|P|/\text{MAX}(|t|)$ teams. In the following implementation (Algorithm 1) I optimize *expected utility*, though another utility function f may be substituted in place.

Algorithm 1 GreedyLocalSearch(RBHG instance B, empty partition π, $\text{MAX}(|t|) = m$)

```
for |P|/m teams do
  select a pivotal agent p
  for m − 1 positions do
    set max index i_MAX to null
    set max score s_MAX to MIN(u_p(r, c)) · |P|
    for |P|/m remaining agents do
      calculate expected utility s'_p for t ∪ p' (in time
      O(|C| · |m|^3))
      if s'_p > s_MAX, set i_MAX = p' and s_MAX = s'_p
    end for
    set t = t ∪ p'
  end for
  set π = π ∪ t
end for
```

Observation 3 *The time complexity of greedy local search is $\mathcal{O}(|P|^2/m \cdot f)$, where f is the running time of the optimization function. In the case of optimizing* expected utility *or* expected stability*, $f = \mathcal{O}(|C| \cdot m^3)$ and the total run time is $\mathcal{O}(|P|^2 \cdot |C| \cdot m^2)$.*

For a special case of RBHG which I call *soul mates RBHG*, greedy local search always returns a perfect partition. For each agent in such an instance, there is a unique set of agents perfectly complimenting its preferences.

Definition 9 *An instance B of* soul mates RBHG *is a 0-1 instance of RBHG which has the property that, for each agent $p \in P$, there is a* unique *set of agents $P' \subseteq P$ such that each* acceptable composition *is also a* stable composition*. I call a set $t = p \cup P'$ having this property* soul mates*.*

Theorem 5 *For any instance B of* soul mates RBHG*, Greedy Local Search always returns a perfect partition.*

Proof Sketch Consider the first iteration of *greedy local search heuristic* on an instance B of *soul mates RBHG*. Select some agent p as the *pivotal agent*. By definition, there is a unique set of agents $P' \subseteq P$ which are *soul mates* with p. When selecting the first agent p' who locally optimizes *expected stability* of $t^p = p \cup p'$, the agent must be some $p' \in P'$.

Observe that, by the distributive property, $p \cup P' = (p \cup p') \cup (P' \cap p')$ given that $p' \in P'$. Therefore, the *soul mates* property is preserved between $p \cup p'$ and $P' \cap p'$. In subsequent iterations, the remaining members of $P' \cap p'$ will be iteratively selected and added to the team t^p until it is completely formed and added to the partition π.

Scraping League of Legends Game Data

For my experiments, I consider a population of League of Legends players where $|P| = 1081$. I considered a set $|R| = 5$ of popular champion roles, *Jungler, AD Carry, Tank, Support,* and *Middle*. These roles were identified as the most common roles of champions (game avatars) selected by the players. I scraped data from the most recent 20 matches for each player $p \in P$ from www.lolking.net, storing 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. Active accounts having played 10 matches within the last 7 days were kept for further scraping, with new matches identified by the match date.

I applied frequent item set mining over team compositions to identify those compositions which were used in at least 3% of matches, and rejected other compositions. This left a set $|C| = 8$ of compositions used in more than 60% of all scraped matches combined.

To compute a player's utilities for role and composition pairs I used the following *wins and losses* utility function.

Definition 10 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 (r, c) than they lost, $u_p(r, c) = -1$ if the agent lost more matches with (r, c) than they won, and $u_p(r, c) = 0$ otherwise.*

I consider an agent to accept *a pair (r, c) if $u_p(r, c) = 1$ and otherwise to* reject *it.*

Testing and Results

Mean results of the experiments are presented in Table 2 while standard deviations are presented in Table 3. I performed matchmaking using "what if" analysis (*W*), greedy voting (*V*) (Spradling et al. 2013), and greedy local search (*L*). I compare the algorithms in terms of run time (in *seconds*) to form the partition π (RT), the percent of teams in π with at least one stable matching (%S), mean expected utility ($\bar{EU}$), median expected utility ($\tilde{EU}$), mean expected stability ($\bar{ES}$), and median expected stability ($\tilde{ES}$). Expected utility ranges from -5 to 5 while expected stability ranges from 0 to 1. I consider 50 of the 52 trials performed, dropping the two with highest and lowest %S for each method.

For "what if" analysis, I consider three different optimization goals; maximizing percent of teams with at least one stable matching (*%S*), maximizing mean expected utility ($\bar{EU}$), and maximizing mean expected stability ($\bar{ES}$). For each optimization goal, "what if" analysis was given a threshold of $\delta = |P|$ (perfect utility 1 for each agent) and a cap of $\Delta = |P|$ partitions. Note that $|P|$ iterations of "what if" analysis on these optimization functions will run in $\mathcal{O}(|P|^2 \cdot |C| \cdot m^2)$. This matches the running time of greedy local search on the same optimization functions.

For greedy local search, I used expected utility as the optimization goal and considered three methods of selecting a pivotal agent: *random pivoting* where an agent is selected i.i.d. from P, *max first pivoting* which pivots around the agent maximizing $q_p = \sum_B u_p(r, c)$, and *min first pivoting* which minimizes $q_p = \sum_B u_p(r, c)$.

Computations were run on a machine using 16 GB of RAM and a 2.50 GHz Intel(R) Core(TM) i7-4710MQ CPU. All algorithms were implemented in Python 3.5.

Testing on Randomized RBHG instances

I tested greedy local search on randomly generated RBHG instances for population sizes from 100 to 1000. The testing showed improved optimization results for the heuristic versus random partitioning as the population size increased. My experiments tested several different formulas for how to randomize the utilities of agents. It became clear that the availability of high quality results depended on how the utility matrix was formed. I saw that the algorithm generally found higher utility solutions on the scraped instances, perhaps because it leveraged the underlying structure of real human preferences.

Table 2: Algorithm mean results over 50 trials

Method	**RT**	**%S**	$\bar{EU}$	$\tilde{EU}$	$\bar{ES}$	$\tilde{ES}$
W (%S)	207.4	22.4%	2.09	2.08	0.03	0
W ($\bar{EU}$)	207.8	16.0	2.13	2.13	0.02	0
W ($\bar{ES}$)	207.8	22.3	2.09	2.09	0.03	0
V	3.8	34.3	2.10	2.13	0.05	0
L (rand)	236.6	55.5	2.75	2.78	0.10	0.13
L (max)	245.9	57.4	2.75	2.8	0.10	0.13
L (min)	230.5	50.5	2.78	2.82	0.09	0.13

Table 3: Algorithm std deviations of results over 50 trials

Method	**RT**	**%S**	$\bar{EU}$	$\tilde{EU}$	$\bar{ES}$	$\tilde{ES}$
W (%S)	4.3	0.9%	0.01	0.06	0.00	0
W ($\bar{EU}$)	4.6	2.3	0.01	0.01	0.00	0
W ($\bar{ES}$)	5.0	0.8	0.01	0.06	0.00	0
V	0.1	0.0	0.00	0	0.00	0
L (rnd)	4.6	1.8	0.02	0.04	0.00	0.01
L (max)	4.8	0.0	0.00	0.00	0.00	0
L (min)	4.8	0.0	0.00	0.00	0.00	0

Rate of growth for "what if" analysis

As part of the "what if" analysis procedure I logged the number of partitions considered at each point that a new local optima was discovered. I observed that the number of new partitions which had to be considered to find an improvement increased by approximately $\mathcal{O}(2^n)$ on these experiments, where n is the number of improvements found so far. Each improvement costs exponentially more time.

Conclusions

Over the League of Legends data set, greedy local search out-performs both greedy voting and "what if" analysis on all optimization measures I considered. As previously observed, the cap of $\Delta = |P|$ partitions for "what if" analysis gives it a similar running time to greedy local search with a slightly smaller constant. In addition to improving expected utility, greedy local search is the only one to achieve a non-zero median expected stability. In its favor, greedy voting is significantly faster and still out-performs "what if" analysis in terms of percent of teams with a stable matching (%S). Greedy voting could used as an alternative for split-second changes to a partition.

On these experiments, greedy local search optimizes expected utility and expected stability at roughly the same

quality whether I used min first or max first pivoting. The difference between the two is no more than the standard deviation of randomized pivoting. Pivot choice shows a more significant variance on the percent of teams with a stable matching (%S). Here, max first pivoting improves by 1.9 percentage points over random pivoting and by 6.9 percentage points over min first pivoting.

The percent of teams which can be stabilized is valuable to optimize when direct role assignment is possible, such as with drones being issued commands by home base. High expected utility and stability are valuable optimization goals when a central authority cannot assume direct control of role matching, such as with human players in an online gaming environment. These experiments provide some positive results for both settings. Even when direct role assignment is not possible or desirable, the optimal role assignments for a partition may be offered as recommendations to the agents once teams have been assigned. In the case of a League of Legends game, this could take the form of suggesting particular champions (avatars) the players have used well in the past and which compliment their team's overall preferences. Software could say, for example, "Your team could use a great Support like you. Why not play Blitzcrank this round?"

Future work will test live matchmaking scenarios where agents are assigned to teams and are alternatively matched to recommended roles or allowed to select roles freely. This user study will test the acceptance of recommendations, the quality of partitioning, and the accuracy of various utility functions for computing role and composition preferences.

References

Alexiou, A., and Doerga, A. 2015. Sprites and stories: The impact of hedonic game elements on perceived learning outcomes. In *Academy of Management Proceedings*, volume 2015, 17225. Academy of Management.

Aziz, H.; Brandt, F.; and Seedig, H. G. 2010. Optimal partitions in additively separable hedonic games. *arXiv preprint arXiv:1005.4540*.

Aziz, H.; Brandt, F.; and Seedig, H. G. 2011. Stable partitions in additively separable hedonic games. In *Proc. AAMAS '11*, 183–190.

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

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

Conitzer, V., and Sandholm, T. 2005. Communication complexity of common voting rules. In *Proceedings of the 6th ACM conference on Electronic commerce*, EC '05, 78–87. New York, NY, USA: ACM.

Darmann, A.; Elkind, E.; Kurz, S.; Lang, J.; Schauer, J.; and Woeginger, G. 2012. Group activity selection problem. In *Internet and Network Economics*. Springer. 156–169.

Diamantoudi, E., and Xue, L. 2003. Farsighted stability in hedonic games. *Social Choice and Welfare* 21(1):39–61.

Drèze, J., and Greenberg, J. 1980. Hedonic coalitions: Optimality and stability. *Econometrica* 48(4):987–1003.

Edmonds, J., and Karp, R. M. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)* 19(2):248–264.

Gamoloco. 2017. Leading gaming content on twitch worldwide in january 2017, by number of hours viewed in millions. Statista - The Statistics Portal. https://www.statista.com/statistics/507786/leading-game-content-twitch-by-number-hours-viewed/.

Guazzone, M.; Anglano, C.; and Sereno, M. 2014. A game-theoretic approach to coalition formation in green cloud federations. In *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, 618–625. IEEE.

Hock, A., and Schoellig, A. P. 2016. Distributed iterative learning control for a team of quadrotors. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, 4640–4646. IEEE.

Kim, S. 2014. An adaptive smart grid management scheme based on the coopetition game model. *ETRI Journal* 36(1):80–88.

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

Peters, D. 2016. Complexity of hedonic games with dichotomous preferences. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence. AAAI Press*.

Saad, W.; Han, Z.; Basar, T.; Hjorungnes, A.; and Song, J. B. 2010. Hedonic coalition formation games for secondary base station cooperation in cognitive radio networks. In *Wireless Communications and Networking Conference (WCNC)*, 1–6.

Saad, W.; Han, Z.; Basar, T.; Debbah, M.; and Hjorungnes, A. 2011a. Hedonic coalition formation for distributed task allocation among wireless agents. In *Proc. IEEE Transactions on Mobile Computing*.

Saad, W.; Han, Z.; Hjørungnes, A.; Niyato, D.; and Hossain, E. 2011b. Coalition formation games for distributed cooperation among roadside units in vehicular networks. *IEEE Journal on Selected Areas in Communications (JSAC), Special issue on Vehicular Communications and Networks*.

Spradling, M., and Goldsmith, J. 2015. Stability in role based hedonic games. In *Proc. FLAIRS-28*. Springer.

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

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

Xu, K.; Wang, K.-C.; Amin, R.; Martin, J.; and Izard, R. 2015. A fast cloud-based network selection scheme using coalition formation games in vehicular networks. *IEEE Transactions on Vehicular Technology* 64(11):5327–5339.