A new model for coalition formation games

Alfred Samman
Louisiana State University and Agricultural and Mechanical College

A NEW MODEL FOR COALITION FORMATION GAMES

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Alfred Samman
B.Sc., University of Botswana, 2003
M.S., Southern University, 2006
August 2012

Acknowledgments

This dissertation would not have been possible without the help and support of numerous people. First and foremost, I would like to thank my advisor Dr. Busch for his patience and guidance. He spent countless hours with me writing and re-writing this dissertation. I would also like to thank Dr. Kannan who helped me see the bigger picture and offered numerous alternative directions whenever my work seemed stagnant. A big thank you goes out to my family, both in the United States, in Botswana and in Ghana, for their support and encouragement. I would like to thank all my friends. Their excitement was contagious. Lastly, all this would not have been possible without this great institution, LSU. Thank you for giving me this tremendous opportunity.

Table of Contents

Acknowledgments ii

Abstract v

I Group Matching Games 1

1 Introduction to Group Matching Games 2
 1.1 Matching Problems 3
 1.2 Coalition Formation 5
 1.3 Set Cover Games 6
 1.4 Set Packing Problem 7
 1.5 Outline 8

2 Definitions for Group Matching Games 9

3 Analysis of Group Matching Games 13
 3.1 Stability 13
 3.2 Price of Anarchy 14
 3.3 Price of Stability 18

4 Group Matching Game with $\lambda > 1$ 24
 4.1 Price of Anarchy for Game with $\lambda > 1$ 24
 4.2 Price of Stability for $\lambda > 1$ 30

5 Group Matching Game with Subset Property 37
 5.1 PoA Analysis 37
 5.2 Lower Bound 39

6 Group Cover Game 40
 6.1 Price of Anarchy for Group Cover Game 40

7 Conclusions to Group Matching Games 42

II Bottleneck Routing Games on Grids 43

8 Introduction to Bottleneck Routing Games on Grids 44
8.1 Contributions 45
8.2 Impact of Games with Small Number of Bends 47
8.3 Related Work 48
8.4 Outline 49

9 Definitions for Bottleneck Routing Games 50

10 Basic Bottleneck Routing Game 52
10.1 Price of Anarchy Analysis for Basic Bottleneck Routing Game 53
10.2 Conclusions 55

11 Channel Routing Games on Grids 57
11.1 Channel Game 57
11.1.1 Price of Anarchy Analysis for Channel Game 58
11.2 Lower Bound 61
11.3 Conclusions 64

12 Split Routing Games on Grids 65
12.1 Split Game 65
12.1.1 Price of Anarchy Analysis for Split Game 66
12.1.2 Split Game in the d-Dimensional Grid 68
12.2 Lower Bound 69
12.3 Conclusions 71

13 Conclusions to Bottleneck Routing Games on Grids 72

References 73

Vita 76

Abstract

Here we present two broad categories of games, namely, *group matching games* and *bottleneck routing games on grids*. Borrowing ideas from coalition formation games, we introduce a new category of games which we call group matching games. We investigate how these games perform when agents are allowed to make selfish decisions that increase their individual payoffs versus when agents act towards the social benefit of the game as a whole. The Price of Anarchy (PoA) and Price of Stability(PoS) metrics are used to quantify these comparisons. We show that the PoA for a group matching game is at most $k\alpha$ and the PoS is at most $\frac{k}{\alpha}$ where k is the maximum size of a group and α is a switching cost. Furthermore we show that the PoA and PoS of the games do not change significantly even if we increase λ, the number of groups that an agent is allowed to join.

We also consider routing games on grid network topologies.The social cost is the worst congestion in any of the network edges (bottleneck congestion). Each player's objective is to find a path that minimizes the bottleneck congestion in its path. We show that the price of anarchy in bottleneck games in grids is proportional to the number of bends β that the paths are allowed to take in the grids' space. We present games where the PoA is $\widetilde{O}(\beta)$. We also give a respective lower bound of $\Omega(\beta)$ which shows that our upper bound is within only a poly-log factor from the best achievable price of anarchy. A significant impact of our analysis is that there exist bottleneck routing games with small number of bends which give a poly-log approximation to the optimal coordinated solution that may use an arbitrary number of bends. To our knowledge, this is the first tight analysis of bottleneck games on grids.

Part I

Group Matching Games

Chapter 1
Introduction to Group Matching Games

Coalition formation games are a class of games in which agents are partitioned and each agent's payoff is dependent on the members of their partition. We borrow some ideas from these games to create group matching games. Our aim in this paper is to investigate a broad range of group matching games from a game theoretic point of view. We analyze how various group matching games behave when individual agents selfishly form matches that maximize their local utilities without considering the needs of the system as a whole. In other words, without a central matchmaker that matches agents together, if agents are allowed to match with other agents based on their local needs, how worse will the utility of the system as a whole be?

The model outlined in this paper has several real world applications. The model can represent a kidney exchange market where agents barter exchange kidneys. A group in this model represents a cycle of agents battering kidneys and the utility of the group is the utility that the agents gain from participating in that group.

Another application is in wireless sensor networks. In this application a group represents a set of agents communicating with each other. It would be ideal for agents close to each other to form a group. This is because the further away agents are from each other the more power is used for communication. Since autonomous wireless agents have a finite supply of power they would want to conserve power by forming clusters of groups in close proximity to one another. The overall social aim is to reduce the average energy consumption. Each agents would in turn want to join local groups so as to minimize their individual power consumption.

The model can also apply to online social networks. Online social networks are often difficult to scale because of the nature of the links between the users. It is difficult to partition and distribute the data because all the users are connected. One solution to this problem is to assign a utility measure to groups of friends. Each user would then decide which group to join based on the utility measures. Once we have a stable assignment of groups we could then put all the data pertaining to a particular group in the same location. This in turn improves performance because related data is in close proximity to each other and so can be retrieved faster.

The list of applications is endless and can include any application that involves grouping entities together and having these entities choose which group to join based on individually perceived utility gain while also considering improving an overall social utility.

In this paper we are primarily concerned with answering the following questions. In a matching game, how worse is the overall social utility if agents behave selfishly versus if they where matched centrally? This is important because we cannot always centrally manage matching games when the number of agents is large. Often these problems become NP-hard problems thus the need to allow agents to form matchings amongst themselves.

The games introduced in this paper are closely related to the matching problem, coalition formation and the set cover problem.

1.1 Matching Problems

There are several classical matching problems, the most common one is the *stable marriage problem* (SMP). In this problem we are presented with an equal number of men and women each with a list of members of the opposite sex that they would like to marry, in order of preference. The solution is a matching of men to women such that no two people of the opposite sex would prefer to be with each other rather than their currently assigned partners. Gale and Shapley outline a solution to this problem that produces a stable matching in polynomial time, i.e. $O(n^2)$ worst case running time.[14]. The *stable roommate problem*

(SRP) is similar to the SMP except that the SRP is not bipartite. That is, we do not have a separate set of men and woman but rather any roommate can be matched with another. Irving et al developed an $O(n^2)$ algorithm to determine if, in a given instance, there is a stable matching and if so what that matching is [16].

The *top trading cycles* (TTC) is another related problem. In this problem each agent has a house that they would like to trade for a better house. Each agent also has an ordered list of houses that they would prefer. The solution to the problem is an assignment of agents to houses in which every agent is assigned to a house that is as good or better than the one they currently have. Shapley and Scarf develop an $O(m)$ algorithm, where m is the total length of the preference list of all the participants, that solves this problem and guarantees that no group of agents can deviate from the given solution to find a better assignment of houses [33].

An important aspect of matching games is the concept of a *matching*. A *matching* is a set of vertex disjoint edges in a graph. That is, a set of edges in which no two edges share a vertex. A maximum matching is the maximum number of such edges in a given graph. Edmonds' matching algorithm is able to find a maximum matching in a given graph in $O(n^4)$ time [13].

More recent work on stable matching include that of Anshelevich et al [2]. Here they investigate how bad bipartite stable matchings are compared with socially optimum matchings and if agents can find stable matchings on their own. A socially optimum matching is one that is orchestrated to maximize a social utility rather than maximizing the utility of the individual agents. Anshelevich et al found that in certain situations two sided stable matching may be arbitrarily worst than a socially optimal matching. To incentivize agents to partake in socially optimum matchings they introduce the concept of *approximate stability*. That is, agents are required to pay a switching cost of α in order to deviate from their current matching. This has been shown the help produce good stable matchings. They primarily investigated two sided matching performed on three different types of graphs. The results are summarized below.

1. Symmetric edge labeled graph. These graphs have undirected edges that are labeled with a weight. The PoA is at most 2. The PoA is at most 2α and the PoS is at most $\frac{2}{\alpha}$ if the graph is α-stable.

2. Vertex labeled graph. These are graphs in which a weight is attached to the vertices. The PoA is at most 2. The PoA is at most $\frac{1+\alpha}{\alpha}$ if the graph is α-stable.

3. Asymmetric edge labeled graph. These are graphs with directional edges that are weighted. The PoA is arbitrarily bad.

Additionally, empirical experiments performed by the authors show that by introducing a modest switching cost the stable matching is quite close to the optimal matching.

1.2 Coalition Formation

Many real world applications require that entities form groups in order to better coordinated their activities. For example, departments within an organization can form groups in other to pool their resources so as to save money. Such application can be modeled as *coalition formation* problems. A *coalition formation game* is a game with finite players in which a feasible allocation is a partition of the agents and an agent's payoff only depends on the members of his/her coalition. Two important notion in coalition formation games are the notion of *anonymity* and *additive separability.* With anonymity all agents are seen as equal and so an agent is not concerned about the identity of the other agents in its coalition. It is only concerned with the number of agents in the coalition. In additive separable games any subset of a coalition provides less payoff, so therefore agents prefer to collate to form a bigger grand coalition.

Sandholm et al [31] enumerate three major activities involved in coalition formation. These are,

1. Coalition structure generation. Agents have to be partitioned into coalitions in which they achieve some kind of payoff. This partition is known as the coalition structure. The task of enumerating possible coalitions and finding the optimal coalition structure in not trivial. Sandholm et al [31] tackle this problem.

2. Solving the optimization problem of each coalition. In certain situations it might be expensive for individual agents to solve a combinatory optimization problem, e.g. task allocation, multi-agent planning and scheduling, and so they form coalitions to solve the problem. Sandholm [32] discusses this further.

3. Dividing the value. This involves how the proceeds generated by the coalition is divided amongst the members of the coalition. Shehory and Kraus [35] discuss methods of distributing payoff.

1.3 Set Cover Games

The set cover problem is a well known NP-complete problem. In a set cover game the goal is to find the minimum number of sets needed to cover all the elements in a universal set given a set of subsets of the universal set. In recent times there has been an emergence of a class of games known as set cover games. These games present a game theoretic formulation of the set cover problem. Leibovic and Willet [20] present such a game. Here the elements in the universal set are represented as agents. These agents then make a decision as to which subset they want to be covered by. Agents prefer to be in a subset which is also chosen by many other agents. The authors discover that the PoA is bounded by $\Theta(\log n)$ which matches the know approximate algorithm for the NP-complete problem. Balcan et al [6] present a slight variation of this game. They introduce a central authority into this game. This central authority knows a good approximation of the optimal solution. It then broadcasts this solution to a small fraction of agents. Agents then make decision based on this. The authors show that again, the game is able to achieve a solution that is $\log n$ factor away from the optimal.

1.4 Set Packing Problem

Another problem closely related to the games that we study in this thesis is the *set packing* problem. This problem is one of the most studied problems in combinatorial optimization. It can simply be described as follows. Given a collection of sets find the maximum number of pairwise disjoint sets in this collection. In the weighted version of this problem all the sets have weights and you are tasked with finding the highest weighted collection of pairwise disjoint sets. When the size of the sets are restricted and cannot exceed a number k then we call this problem the k-set packing problem.

Finding the optimal solution to the k-set packing problem has been shown to be NP-hard for any $k \geq 3$ [1]. Furthermore, if $k = 3$ the problem becomes APX-hard [17]. Due to this fact most of the work being done on this problem involves finding good heuristics that give good approximations to the optimal solution. One such heuristic is a greedy algorithm that iteratively adds an arbitrary set and removes all sets intersecting with it. We continue doing this until there are no more sets to be added. It is shown that this algorithm provides a k approximation to the optimal solution.

Another approach is the local search heuristic. In this approach we attempt to replace a small subset of the solution with a collection that is of greater weight [11, 3]. Arkin and Hassin [3] show that local search results in an approximation of $k - 1 + \epsilon$ where $\epsilon > 0$. Bafna et al [5] also collaborate this assertion.

Chandra and Halldórsson [11] combine the greedy algorithm and the local search heuristic to provide a better approximation. They first start with a greedy solution and then repeatedly choose the best possible local improvement. This approach produces a $2(k + 1)/3$ approximation, which is asymptotically tight.

1.5 Outline

The organization of the rest of this thesis is as follows. In Chapter 2 we outline the basic definitions in group matching games. In Chapter 3 we analyze group matching games. We give PoA and PoS bounds. In Chapter 4 we provide an analysis of group matching games with λ greater than 1. We also outline some algorithms that produce stable matching for these types of games. In Chapter 6 we discuss group cover games and we provide a PoA bound for these types of games.

Chapter 2
Definitions for Group Matching Games

We consider games involving agents which are based on the classic set packing problem. We will first introduce some basic terminologies from the set packing problem which we will extend for the games that we will describe below.

We are given a collection of sets $\mathcal{S} = \{g_1, g_2, \ldots, g_m\}$, where each g_i, $1 \le i \le m$, is a set of elements from the universe $N = \{1, \ldots, n\}$. We will refer to the elements of N as *agents*. There is a weight function $w : \mathcal{S} \to \mathcal{R}^+$ which assigns a positive weight $w(g)$ to each set $g \in \mathcal{S}$. A *disjoint sub-collection* $\mathbf{M} \subseteq \mathcal{S}$ has the property that for every $1 \le i, j \le m$, $i \ne j$, $g_i \cap g_j = \emptyset$. For brevity we will refer to a disjoint sub-collection as a *matching*. In a set packing problem the goal is find a matching $\mathbf{M}$ which maximizes $\sum_{g_i \in \mathbf{M}} w(g_i)$.

We consider group matching games where each game is expressed as a tuple $G = \langle N, \mathcal{S}, (c_i), (S_i), (u_i) \rangle$, where for each agent i we have a fixed weight $c_i > 0$ (we denote $(c_i) = (c_1, \ldots, c_n)$), a *strategy set* $S_i \subseteq \mathcal{S}$ such that for each $g \in S_i$, $i \in g$, and a *utility function* u_i defined below.

In the game every player chooses a group in its strategy set. An *allocation* is a tuple $a = (a_1, \cdots, a_n)$, where $a_i \in S_i \cup \{\emptyset\}$, which represents the groups chosen by the agents. Note that in an allocation an agent i may not choose any group in which case $a_i = \emptyset$. An allocation is *valid* if for each $g \in \bigcup_i a_i$, $1 \le i \le n$, all the agents (elements) in g must have chosen g in the allocation, namely, for each agent $j \in g$, $a_j = g$. The groups in a valid allocation form a *matching*. In other words a matching is a set of all the elements in a valid allocation. Sometimes we refer to a valid allocation as a matching and vice versa.

The utility of an agent i is a function that accepts an allocation and returns a real number, namely, $u_i : \times_{i=1}^{n} (S_i \cup \{\emptyset\}) \to \mathcal{R}$. More specifically, we set the utility function $u_i(a) = w(a_i)$,

TABLE 2.1. An example demonstrating the PoA of a game

$N = \{1,2,3,4,5\}$				
$g_1 = \{1,2\}, g_2 = \{2,3\}, g_3 = \{3,4,5\}$				
$\mathcal{S} = \{g_1, g_2, g_3\}$				
$S_1 = \{g_1\}, S_2 = \{g_1, g_2\}, S_3 = \{g_2, g_3\}, S_4 = \{g_3\}, S_5 = \{g_3\}$				
$a = (a_1, a_2, a_3, a_4, a_5) = (\emptyset, g_2, g_2, \emptyset, \emptyset),\ a^* = (g_1, g_1, g_3, g_3, g_3)$				
$\mathbf{M} = \{g_2\}, \mathbf{M}^* = \{g_1, g_3\}$				
$c_1 = 2,$	$c_2 = 3,$	$c_3 = 4,$	$c_4 = 1,$	$c_5 = 1$
$u_1(\mathbf{M}^*) = 5,$	$u_2(\mathbf{M}^*) = 5,$	$u_3(\mathbf{M}^*) = 6,$	$u_4(\mathbf{M}^*) = 6,$	$u_5(\mathbf{M}^*) = 6$
$u_1(\mathbf{M}) = 0,$	$u_2(\mathbf{M}) = 7,$	$u_3(\mathbf{M}) = 7,$	$u_4(\mathbf{M}) = 0,$	$u_5(\mathbf{M}) = 0$
$w(g_1) = 5, w(g_2) = 7, w(g_3) = 6$				
$SU(\mathbf{M}^*) = 11, SU(\mathbf{M}) = 7$				
$PoA = \frac{11}{7}$				

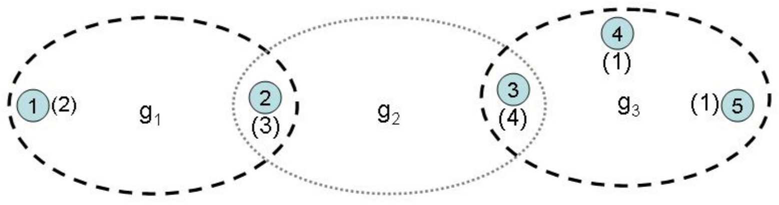


FIGURE 2.1. An example of a group matching game. The small circles represent agents and the larger circles with broken outlines represent groups. The numbers in parenthesis represent the weights of the agents. Groups g_1, g_3 form a optimal matching and group g_2 forms the stable matching.

where $w(g) = \sum_{i \in g} c_i$ and $w(\emptyset) = 0$. For a matching $\mathbf{M}$ we will also use the notation $u_i(\mathbf{M})$ to denote $u_i(a)$ for the respective allocation a. Table 2.1 and Figure 2.1 show an example of a group matching game.

We say that a group g_1 is *adjacent* to g_2 if they have some agents in common, i.e. $g_1 \cap g_2 \neq \emptyset$. The *neighborhood* of a group g, denoted $X(g)$, is the set of all groups *adjacent* to g, namely, $X(g) = \{g_i : g_i \cap g \neq \emptyset\}$.

An agent's goal is to maximize its utility. Note that, we are only interested in valid allocations (matchings). A matching $\mathbf{M}$ is *stable* if players cannot improve their utilities by changing their current group choices. In other words, in a stable matching, no group g' in $\mathcal{S} \setminus \mathbf{M}$ can be formed by agents deflecting from their currently assigned groups because at least one member of g' cannot improve its utility. In particular, let $\mathbf{M}' = (\mathbf{M} \cup \{g'\}) \setminus \{g \in \mathbf{M} : g \cap g' \neq \emptyset\}$ be the matching where g' has been formed by agents deflecting from their group choices in $\mathbf{M}$. There is a player $i \in g'$ such that the utility of i in $\mathbf{M}'$ is not greater than in $\mathbf{M}$. Therefore, for each $g' \in \mathcal{S} \setminus \mathbf{M}$, there is $i \in g'$, such that $u_i(\mathbf{M}) = w(g) \geq w(g') = u_i(\mathbf{M}')$, $i \in g \in \mathbf{M}$.

We now extend the definition of a stable matching to include the notion of an *α-stable matching* for any $\alpha \geq 1$. A matching $\mathbf{M}$ is α-stable if players cannot improve their utilities by a factor of α by changing their current group choices. In other words, no group g' in $\mathcal{S} \setminus \mathbf{M}$ can be formed by agents deflecting from their currently assigned groups because at least one member of g' cannot improve its utility by more than a factor of α by deflecting. In particular, let $\mathbf{M}' = (\mathbf{M} \cup \{g'\}) \setminus \{g \in \mathbf{M} : g \cap g' \neq \emptyset\}$ be the matching where g' has been formed by agents deflecting from their group choices in $\mathbf{M}$. There is a player $i \in g'$ such that the utility of i in $\mathbf{M}'$ is not more than a factor of α times greater than in $\mathbf{M}$. Therefore, for each $g' \in \mathcal{S} \setminus \mathbf{M}$, there is $i \in g'$, such that $\alpha u_i(\mathbf{M}) = w(g) \geq w(g') = u_i(\mathbf{M}')$, $i \in g \in \mathbf{M}$. If we do not explicitly specify the value of α then it is assumed that $\alpha = 1$.

We define the *social utility* of a matching $\mathbf{M}$ to be $SU(\mathbf{M}) = \sum_{g \in \mathbf{M}} w(g)$. A matching $\mathbf{M}^*$ is *optimal* if it has the largest social utility of all the possible matchings.

Let Q denote all possible stable matchings in the game. Note that Q is finite. The *Price of Anarchy* of the game G is $PoA = \min\limits_{\mathbf{M} \in Q} \frac{SU(\mathbf{M}^*)}{SU(\mathbf{M})}$. In other words the PoA is the ratio of the *optimal matching* to the worst *stable matching*. Another related metric is the *Price of Stability* (PoS) which is defined as $PoS = \max\limits_{\mathbf{M} \in Q} \frac{SU(\mathbf{M}^*)}{SU(\mathbf{M})}$. In other words, it is the ratio of the *optimal matching* to the best *stable matching*.

We can easily extend the definitions of PoA and PoS to α-stable matchings by considering Q to include all such matchings.

Chapter 3
Analysis of Group Matching Games

We analyze group matching games. We first show that there exist stable matchings and then we analyze the price of anarchy and the price of stability.

3.1 Stability

We present Algorithm 1 which computes a stable matching (1-stable matching) for any game G. The algorithm begins by ordering all the groups in the game in decreasing order of group weight. It then iterates through the groups in this order and decides whether to add each group to the resulting set of groups which will be returned as output. A new group is added only if all of its members are not already in the current set of groups. The resulting output is a stable matching.

Algorithm 1: Algorithm to find a stable matching in a group matching game

```
input  : A group matching game $G = \langle N, \mathcal{S}, (c_i), (S_i), (u_i) \rangle$
output: A set of groups $\mathbf{M}$
1 $Z = (g_1, g_2, \ldots, g_{|\mathcal{S}|})$ is a sorted list of groups were $w(g_i) \geq w(g_j), i > j, g_i, g_j \in \mathcal{S}$;
2 $\mathbf{M} \leftarrow \emptyset$;
3 for $i = 1, \ldots, |\mathcal{S}|$ do
4     if all members of $g_i$ are not in any group in $\mathbf{M}$ then
5         $\mathbf{M} \leftarrow \mathbf{M} \cup g_i$;
6 return $\mathbf{M}$
```

Example of Algorithm 1

Figure 3.1 shows an example demonstrating how Algorithm 1 operates. We say that an agent is free if a group in which it is a member of has not been added to the matching $\mathbf{M}$. The algorithm examines the groups in decreasing order of group weight.

In the first iteration of the algorithm, g_2 is added to the matching $\mathbf{M}$ because its agents 2 and 3 are free.

In the next iteration g_1 is examined. It is not included in the matching because one of its agents, 2 is not free.

In the third iteration g_3 is not included in $\mathbf{M}$ because agent 3 is not free.

Finally, g_4 is added to $\mathbf{M}$ because all its agents are free.

Therefore the final matching included groups g_1 and g_4. Note that this matching is stable because no group outside of g_1 and g_4 can be formed without some agent's utility reducing. Remember that an agent's utility is its group weight in a particular matching.

Theorem 3.1.1 *Given a game G, Algorithm 1 returns a stable matching.*

Proof: Let $\mathbf{M}$ be the output of then algorithm. In $\mathbf{M}$ an agent is added to at most one group. This is because when its group is added to $\mathbf{M}$ all other groups that it belongs to are not eligible to be added to $\mathbf{M}$ in subsequent iterations. Therefore, $\mathbf{M}$ is a matching.

Next we will prove the $\mathbf{M}$ is stable. Suppose to the contrary that $\mathbf{M}$ is unstable. This means that there is a group $g \in \mathcal{S} \setminus \mathbf{M}$ causing an instability such that $w(g) > w(g')$ for all $i \in g \cap g'$, $g' \in X(g)$ and $g' \in \mathbf{M}$. Since g' is in $\mathbf{M}$ then from line 4 we have it that, $w(g') \geq w(g'')$ for all $g'' \in X(g')$. This is a contradiction because $g \in X(g')$ but $w(g) > w(g')$. ■

3.2 Price of Anarchy

Consider game G where the maximum group size is k. We provide a *PoA* upper bound for α-stable matchings where the bound is expressed in terms of α and k.

Theorem 3.2.1 *The PoA of a game G for α-stable matchings is αk where k is the maximum group size, for any $\alpha, k \geq 1$.*

Proof: Let $\mathbf{M}^*$ be the optimal matching and $\mathbf{M}$ be an α-stable matching. We proceed by setting an upper-bound for $SU(\mathbf{M}^*)$ with respect to $SU(\mathbf{M})$ which will give the *PoA* bound.

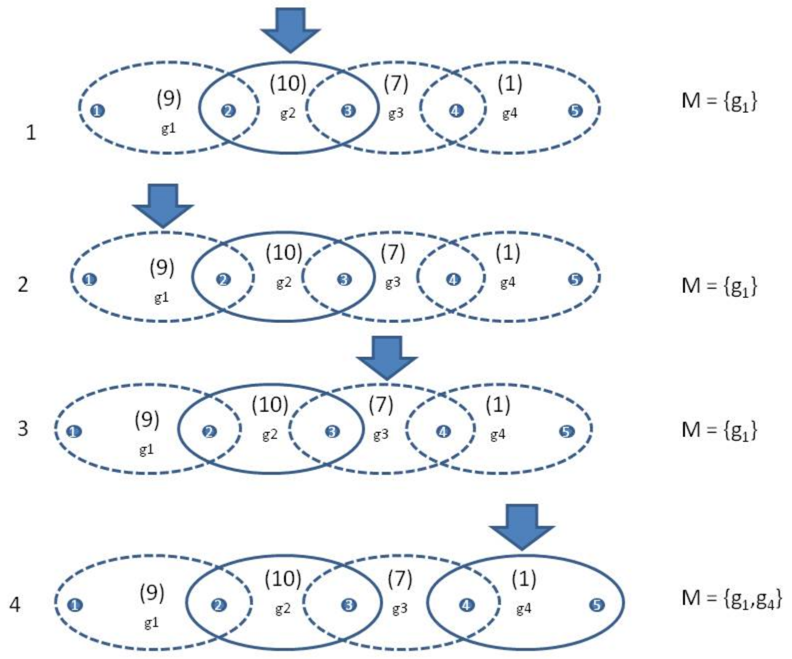


FIGURE 3.1. An example demonstrating how Algorithm 1 operates. The smaller circles represent agents, the larger circles with broken outlines represent groups. The numbers in parenthesis indicates the group's weight.

Let $g^* \in \mathbf{M}^* \setminus \mathbf{M}$, then there exist a $\tilde{g}$, where $\tilde{g} \in X(g^*)$ and $\tilde{g} \in \mathbf{M} \setminus \mathbf{M}^*$, such that,

$$\alpha w(\tilde{g}) \geq w(g^*), \tag{3.1}$$

since $\mathbf{M}$ is an α-stable matching, at least one agent $i \in g \cap g^*$ gets a utility that is not more than α times worse in $\mathbf{M}$ than in $\mathbf{M}^*$.

Therefore, we can map every group $g' \in \mathbf{M}^* \setminus \mathbf{M}$ to a group $g \in \mathbf{M} \setminus \mathbf{M}^*$ such that $w(g') \leq \alpha w(g)$. For $g \in \mathbf{M} \setminus \mathbf{M}^*$, let $F_g \subseteq X(g)$ be the set of groups in $\mathbf{M}^* \setminus \mathbf{M}$ which map to g. Note that,

$$\bigcup_{g \in \mathbf{M} \setminus \mathbf{M}^*} F_g = \mathbf{M}^* \setminus \mathbf{M}. \tag{3.2}$$

A group $g \in \mathbf{M} \setminus \mathbf{M}^*$ can be adjacent to at most k groups in F_g, and therefore $|F_g| \leq k$. From Inequality 3.1 we have,

$$kw(g) \geq w(g)|F_g| \geq \sum_{g' \in F_g} \frac{w(g')}{\alpha}.$$

Therefore,

$$w(g) \geq \frac{1}{k\alpha} \sum_{g' \in F_g} w(g'). \tag{3.3}$$

Thus,

$$\begin{aligned} SU(\mathbf{M} \setminus \mathbf{M}^*) &= \sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} w(g) \\ &\geq \frac{1}{k\alpha} \sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} \sum_{g' \in F_g} w(g') \quad \textit{using Inequality } (3.3) \\ &\geq \frac{1}{k\alpha} \sum_{g' \in \mathbf{M}^* \setminus \mathbf{M}} w(g') \quad \textit{using Inequality } (3.2) \\ &= \frac{1}{k\alpha} SU(\mathbf{M}^* \setminus \mathbf{M}). \end{aligned} \tag{3.4}$$

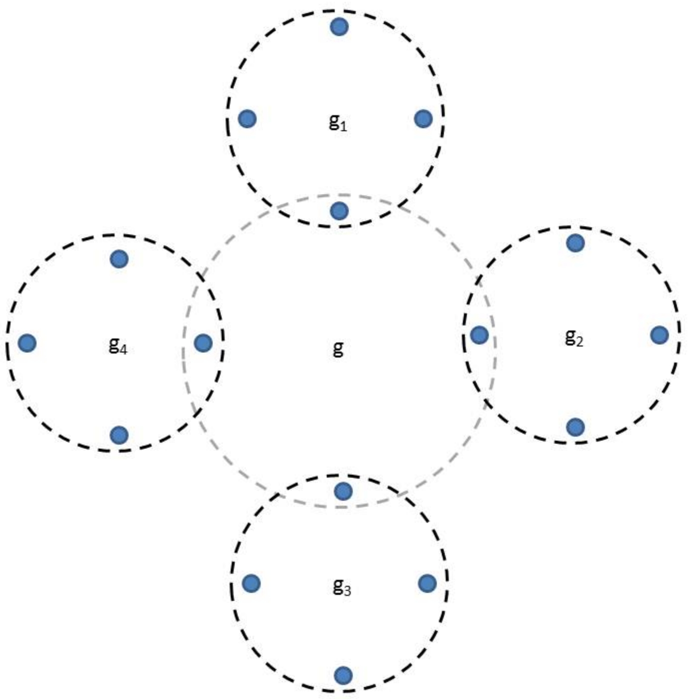


FIGURE 3.2. The small circles represent agents and the larger circles with broken outlines represent groups. Groups g_1, g_2, g_3, g_4 each shares an agent with g. The groups all have size k.

Since $\mathbf{M} = (\mathbf{M} \cap \mathbf{M}^*) \cup (\mathbf{M} \setminus \mathbf{M}^*)$ and $\mathbf{M} \cap \mathbf{M}^*$ is disjoint with $\mathbf{M} \setminus \mathbf{M}^*$, we get

$$\begin{aligned} SU(\mathbf{M}) &= SU(\mathbf{M} \cap \mathbf{M}^*) + SU(\mathbf{M} \setminus \mathbf{M}^*) \\ &\geq SU(\mathbf{M} \cap \mathbf{M}^*) + \frac{1}{k\alpha} SU(\mathbf{M}^* \setminus \mathbf{M}) \quad \textit{using Inequality}\ (3.4) \\ &\geq \frac{1}{k\alpha} \left(SU(\mathbf{M} \cap \mathbf{M}^*) + SU(\mathbf{M}^* \setminus \mathbf{M}) \right) \\ &= \frac{1}{k\alpha} SU(\mathbf{M}^*). \end{aligned} \tag{3.5}$$

Since $\mathbf{M}$ is an arbitrary stable matching, we get $PoA \leq \alpha k$. ■

Theorem 3.2.2 *For any $\alpha, k \geq 1$, there is a game G with maximum group size k such that the $PoA \geq \alpha k$ for α-stable matchings.*

Proof: We define a game G with $\mathcal{S} = \{g, g_1, \ldots, g_k\}$ such that $g_i \cap g_j = \emptyset$ for $i \neq j$ and g shares exactly one agent with each g_i, that is $|g \cap g_i| = 1$ (see Figure 3.2). Therefore, the group g is adjacent to k other groups. Every agent $i \in g$ has weight $c_i = \frac{1}{k}$ and for every agent $j \in (g_i \setminus g)$ has weight $c_j = \frac{k\alpha - 1}{k(k-1)}$. Thus the weights of the groups are,

$$w(g) = \sum_{i=1}^{k} \frac{1}{k} = 1$$

and

$$w(g_i) = \frac{1}{k} + \sum_{i=1}^{k-1} \frac{k\alpha - 1}{k(k-1)} = \alpha.$$

The optimal matching is $\mathbf{M}^* = \{g_1, \ldots, g_k\}$ with $SU(\mathbf{M}^*) = \sum_{i=1}^{k} w(g_i) = k\alpha$. The matching $\mathbf{M} = \{g\}$ is stable since each agent $p \in g_i \cap g$ will choose g because it does not improve it utility by moving to g_i, since $u_p(\mathbf{M}^*) = w(g_i) = \alpha = \alpha w(g) = \alpha u_p(\mathbf{M})$. The only other stable matching is $\mathbf{M}^*$ which has a higher social utility than $\mathbf{M}$. Therefore,

$$\begin{aligned} PoA &= SU(\mathbf{M}^*)/SU(\mathbf{M}) \\ &= k\alpha. \end{aligned}$$

■

3.3 Price of Stability

We present Algorithm 2 which computes an α-stable matching with a low social utility given a group matching game. Firstly, it arranges all the groups in the game in descending order of group weight in Line 1. It then initializes the current set of groups $\mathbf{M}$ to be the set of groups that form an optimal matching (Line 2). It then iterates through these groups in this descending order of group weights checking whether the group currently being examined is a factor of α better than all its neighboring groups (Line 4). If this criteria is met then the neighboring groups are all replaced by the currently examined group in the current set of groups (Line 5 and 6). The final output of the algorithm is an α-stale matching as we prove below. In addition the social utility $\frac{k}{\alpha}$ worse than the optimal social matching.

Algorithm 2: An algorithm to produce an α-stable matching

```
input  : A group matching game $G = \langle N, \mathcal{S}, (c_i), (S_i), (u_i) \rangle$
output: A set of groups $\mathbf{M}$
1 $Z = (g_1, g_2, \ldots, g_{|\mathcal{S}|})$ is a sorted list of groups were $w(g_i) > w(g_j), i > j, g_i, g_j \in \mathcal{S}$;
2 $\mathbf{M} \leftarrow \mathbf{M}^*$;
3 for $i = 1, \ldots, |\mathcal{S}|$ do
4     if $\frac{w(g_i)}{\alpha} > w(g')$ for all $g' \in X(g_i)$ then
5         $\mathbf{M} \leftarrow \mathbf{M} \setminus g'$, for all $g' \in X(g)$ ;  // remove all groups adjacent to $g_i$ from $\mathbf{M}$
6         $\mathbf{M} \leftarrow \mathbf{M} \cup g_i$;
7 return $\mathbf{M}$;
```

Example of Algorithm 2

Figure 7 shows an example of Algorithm 2 in operation. The algorithm begins with $\mathbf{M}$ being the optimal matching, it then examines the groups in decreasing order of weight. In this example $k = 3$ and $\alpha = 2$.

In the first iteration the algorithm examines group g_1. Since its weight is a factor of α greater than its neighbors ($w(g_1)/2 > w(g_2)$, $w(g_1)/2 > w(g_3)$ and $w(g_1)/2 > w(g_4)$), it is added to $\mathbf{M}$ and its neighbors g_2, g_3 and g_4 are removed from $\mathbf{M}$.

The group g_6 is then examined. Since it is not a factor of α greater then its neighbors, it is not removed from the matching $\mathbf{M}$.

Theorem 3.3.1 *Algorithm 2 produces an α-stable matching.*

Proof: Suppose to the contrary that the matching $\mathbf{M}$ returned by the algorithm is unstable. This means that there is a group $g \notin \mathbf{M}$ causing an instability such that $\alpha w(g) > w(g')$ for all $i \in g \cap g'$, $g' \in X(g)$ and $g' \in \mathbf{M}$. Since g' is in $\mathbf{M}$ then from Line 4 we can say that, $w(g')/\alpha > w(g'')$ for all $g'' \in X(g')$. This is a contradiction because $g \in X(g')$ but $w(g) > \frac{w(g')}{\alpha}$. ∎

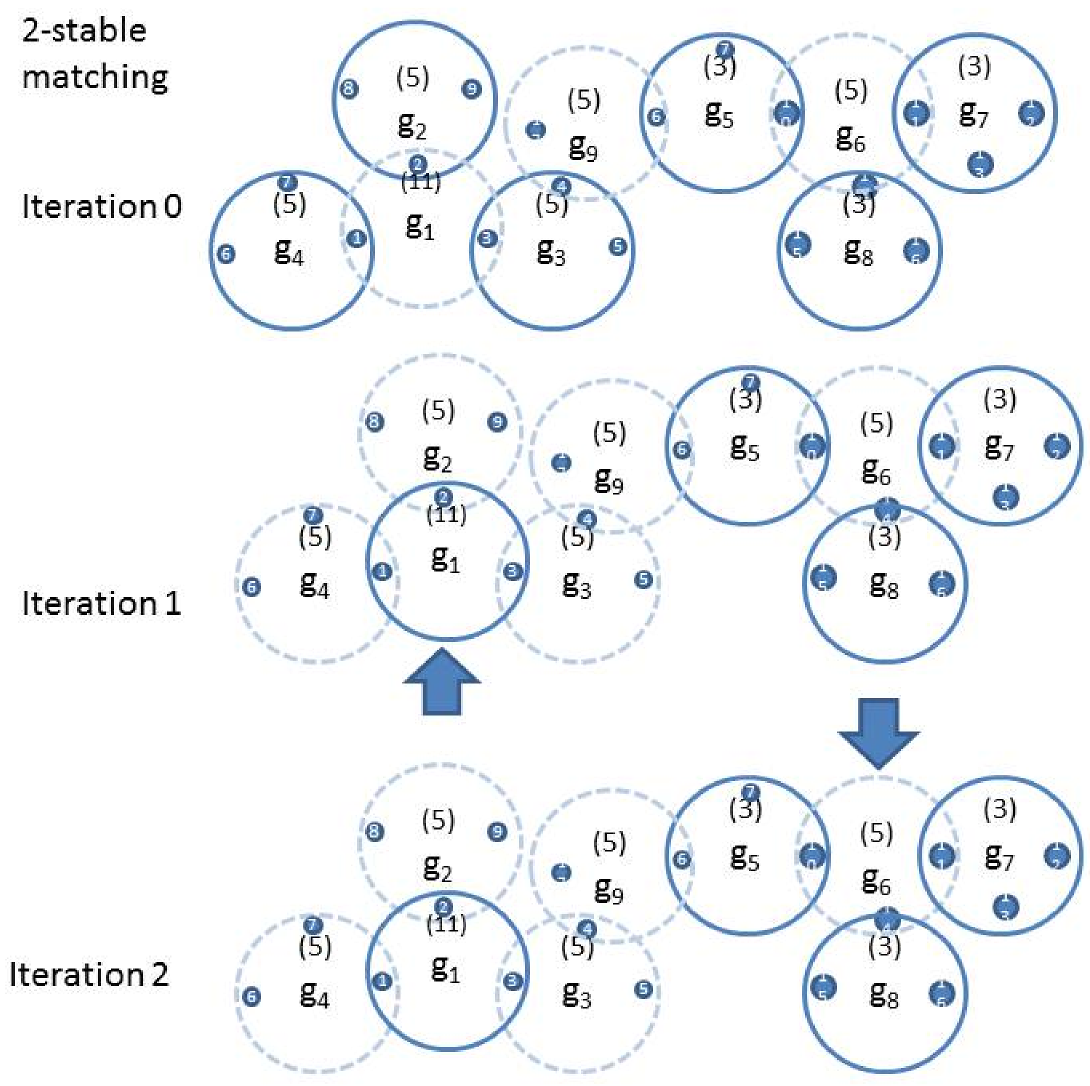


FIGURE 3.3. An example demonstrating how Algorithm 2 operates. The smaller circles represent agents, the larger circles with broken outlines represent groups. The numbers in parenthesis indicates the group's weight.

Theorem 3.3.2 *Every group matching game has PoS $\leq k/\alpha$ for α-stable matchings where k is the maximum group size.*

Proof: Let $\mathbf{M}^*$ be the optimal matching and $\mathbf{M}$ be the α-stable matching returned by Algorithm 2. Initially when the algorithm starts $\mathbf{M}^* = \mathbf{M}$. During the execution of the algorithm several groups from $\mathbf{M}^*$ are removed from $\mathbf{M}$. Let $g^* \in \mathbf{M}^* \setminus \mathbf{M}$ be one such group. Then, from Line 4 there exist a $\tilde{g}$, where $\tilde{g} \in X(g^*)$ and $\tilde{g} \in \mathbf{M} \setminus \mathbf{M}^*$ such that,

$$\alpha w(g^*) < w(\tilde{g}). \tag{3.6}$$

Therefore, we can map every group $g' \in \mathbf{M}^* \setminus \mathbf{M}$ to a group $g \in \mathbf{M} \setminus \mathbf{M}^*$ such that $w(g) > \alpha w(g')$. For $g \in \mathbf{M} \setminus \mathbf{M}^*$, let $F_g \subseteq X(g)$ be the set of groups in $\mathbf{M}^* \setminus \mathbf{M}$ which map to g. Note that,

$$\bigcup_{g \in \mathbf{M} \setminus \mathbf{M}^*} F_g = \mathbf{M}^* \setminus \mathbf{M}. \tag{3.7}$$

A group $g \in \mathbf{M} \setminus \mathbf{M}^*$ can be adjacent to at most k groups in F_g, and therefore $|F_g| \leq k$. From Inequality 4.1 we have,

$$kw(g) \geq w(g)|F_g| > \sum_{g' \in F_g} \alpha w(g').$$

Therefore,

$$w(g) \geq \frac{\alpha}{k} \sum_{g' \in F_g} w(g'). \tag{3.8}$$

Thus,

$$\begin{aligned} SU(\mathbf{M} \setminus \mathbf{M}^*) &= \sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} w(g) \\ &\geq \frac{\alpha}{k} \sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} \sum_{g' \in F_g} w(g') \quad \textit{using Inequality}\ (4.3) \\ &\geq \frac{\alpha}{k} \sum_{g' \in \mathbf{M}^* \setminus \mathbf{M}} w(g') \quad \textit{using Inequality}\ (4.2) \\ &= \frac{\alpha}{k} SU(\mathbf{M}^* \setminus \mathbf{M}). \end{aligned} \tag{3.9}$$

Since $\mathbf{M} = (\mathbf{M} \cap \mathbf{M}^*) \cup (\mathbf{M} \setminus \mathbf{M}^*)$ and $\mathbf{M} \cap \mathbf{M}^*$ is disjoint with $\mathbf{M} \setminus \mathbf{M}^*$, we get

$$\begin{aligned} SU(\mathbf{M}) &= SU(\mathbf{M} \cap \mathbf{M}^*) + SU(\mathbf{M} \setminus \mathbf{M}^*) \\ &\geq SU(\mathbf{M} \cap \mathbf{M}^*) + \frac{\alpha}{k} SU(\mathbf{M}^* \setminus \mathbf{M}) \quad \textit{using Inequality}\ (3.9) \\ &\geq \frac{\alpha}{k} \left(SU(\mathbf{M} \cap \mathbf{M}^*) + SU(\mathbf{M}^* \setminus \mathbf{M}) \right) \\ &= \frac{\alpha}{k} SU(\mathbf{M}^*). \end{aligned} \tag{3.10}$$

Consequently, we get $PoS \leq \frac{k}{\alpha}$.

■

Theorem 3.3.3 *There is a game G with an α-stable matching such that $PoS \geq k/\alpha(1+\epsilon)$ for $k \geq \alpha$ for an arbitrarily small ϵ.*

Proof: We define a game G with $\mathcal{S} = \{g, g_1, \cdots, g_k\}$ such that $g_i \cap g_j = \emptyset$ for $i \neq j$ and g shares exactly one agent with each g_i, that is $|g \cap g_i| = 1$ (see Figure 3.2). Therefore, the group g is adjacent to k other groups. Every agent $i \in g$ has weight $c_i = \frac{1+\epsilon}{k}$ (where ϵ is a very small number) and for every agent $j \in (g_i \setminus g)$ has weight $c_j = \frac{k-\alpha}{k\alpha(k-1)}$. Thus the weights of the groups are,

$$\begin{aligned} w(g) &= \sum_{i=1}^{k} \frac{1+\epsilon}{k} \\ &= 1+\epsilon, \end{aligned}$$

and

$$\begin{aligned} w(g_i) &= \frac{1}{k} + \sum_{i=1}^{k-1} \frac{k-\alpha}{k\alpha(k-1)} \\ &= \frac{1}{\alpha}. \end{aligned}$$

The game has two matchings, the optimal matching $\mathbf{M}^* = \{g_1, \cdots g_k\}$ and the stable matching $\mathbf{M} = \{g\}$. All agents $p \in g_i \cap g$ will choose g because $u_p(\mathbf{M}^*) < \alpha u_p(\mathbf{M})$, therefore the social welfare in the stable matching will be,

$$SU(\mathbf{M}) = w(g) = 1.$$

If however all the agents choose g_i then,

$$\begin{aligned} SU(\mathbf{M}^*) &= \sum_{i=1}^{k} w(g_i) \\ &= \frac{k}{\alpha}. \end{aligned}$$

Since there is no other stable matching other than $\mathbf{M}$,

$$\begin{aligned} PoS &= SU(\mathbf{M}^*)/SU(\mathbf{M}) \\ &= \frac{k}{\alpha(1+\epsilon)}. \end{aligned}$$

■

Theorem 3.3.4 *For any $x \geq 0$, each $(k+x)$-stable matching is also a k-stable matching, where $k \geq 1$ is the largest group size.*

Proof: Consider a $k+x$-stable matching $\mathbf{M}$. Suppose that $g \in S \setminus \mathbf{M}$. Let $\mathbf{M}'$ be the matching which is obtained from $\mathbf{M}$ by adding g and removing all the members of $X(g)$. Let $i \in g$ be the agent with the largest c_i among all the other agents in g. We have that $c_i \geq w(g)/k$. If $i \in g' \in \mathbf{M}$, then clearly $w(g') \geq c_i \geq w(g)/k$. Then $u_i(\mathbf{M}') = w(g) \leq kw(g') = ku_i(\mathbf{M})$. Consequently, agent i has no incentive to switch to g since there is no improvement by more than a factor of k in its utility. Thus, for each group in $S \setminus \mathbf{M}$ there is an agent which cannot improve its utility by more than a factor of k by switching to that group. Therefore, $\mathbf{M}$ is a k-stable matching. ■

Chapter 4
Group Matching Game with $\lambda > 1$

In this section we introduce the parameter λ which is the maximum number of groups that an agent can be a member of. For these games we extend our original definitions of stability and matching. An *allocation* is a tuple $a = (a_1, \ldots, a_n)$, where $a_i \subseteq S_i \cup \emptyset$, which represents the groups chosen by the agents. Note that in an allocation an agent i may choose up to λ groups and so $|a_i| \leq k$ or not choose any group at all in which case $a_i = \emptyset$. An allocation is *valid* if for each $g \in \bigcup_i a_i$, $1 \leq i \leq n$, all the agents (elements) in g must have chosen g in the allocation, namely, for each agent $j \in g$, $a_j = g$. Once again, the groups in a valid allocation form a *matching*. In other words a matching is a set of all the elements in a valid allocation. Sometimes we refer to a valid allocation as a matching and vice versa.

We define q_i to be the number of groups that agent i has currently chosen. Let $\mathbf{M}$ be a stable matching and $\mathbf{M'} = (\mathbf{M} \cup \{g'\}) \setminus \{g \in \mathbf{M} : g \cap g' \neq \emptyset\}$ be the matching where g' has been formed by agents deflecting from their group choices in $\mathbf{M}$. There is a player $i \in g'$ which has already met its quota of groups it can choose and the utility of i in $\mathbf{M'}$ is not greater than in $\mathbf{M}$. Therefore, for each $g' \in \mathcal{S} \setminus \mathbf{M}$, there is $i \in g'$, such that $u_i(\mathbf{M}) = w(g) \geq w(g') = u_i(\mathbf{M'})$, $i \in g \in \mathbf{M}$ and $q_i = \lambda$.

4.1 Price of Anarchy for Game with $\lambda > 1$

We describe Algorithm 3 which given a game G produces a stable matching. The groups in the game are sorted into descending order of weight. Each agent is given a quota. It represents the number of groups that an agent has joined. Anytime an agent's group is added to $\mathbf{M}$, the current set of groups, its quota is increased. An agent cannot join more than λ groups. Each group is then examined in this descending order of weight. If all the members of the group currently being examined have not met their quota (Line 5) then the group in added to the current set of groups (Line 7) and the quota for each member of the group is increased. If any group does not meet this criteria it is not added to be current set of groups. At the end of the execution of the algorithm the set of groups is returned as a matching.

Algorithm 3: Algorithm to find a stable matching in a simple group packing game with $\lambda > 1$

```
input  : A group matching game $G = \langle N, \mathcal{S}, (c_i), (S_i), (u_i) \rangle$
output: A set of groups $\mathbf{M}$
1 $Z = (g_1, g_2, \ldots, g_{|\mathcal{S}|})$ is a sorted list of groups were $w(g_m) > w(g_n), m > n, g_m, g_n \in \mathcal{S}$;
2 $\mathbf{M} \leftarrow \emptyset$;
3 For each $i \in N, q_i \leftarrow 1$ ;     // each agent $i$ has a quota $q_i$. We initialize the quotas to 1
4 for $j = 1, \cdots, |\mathcal{S}|$ do
5     if $q_i < \lambda$ for all $i \in g_j$ then
6         $q_i \leftarrow q_i + 1$;
7         $\mathbf{M} \leftarrow \mathbf{M} \cup g_j$;
8 return $\mathbf{M}$;
```

Example of Algorithm 3

Figure 7 shows an example of Algorithm 3 in operation. It examines the groups in decreasing order of weight. In this example $k = 3$ and $\lambda = 2$.

In the first iteration the algorithm examines group g_1. The quotas for the members of the group, agents 1, 2 and 3 are increased by one and g_1 is added to $\mathbf{M}$.

In the next iteration group g_2 is examined, the quotas for agents 1, 2 and 4 are increased by one and g_2 is added to $\mathbf{M}$.

In the proceeding iterations, group g_3, g_4 and then g_5 is examined. These groups are not added to $\mathbf{M}$ because agent 1 and 2 have already met their quota of 2.

Note that at the end of the execution of the algorithm, $\mathbf{M}$ is stable.

Theorem 4.1.1 *The matching returned by Algorithm 3 is stable.*

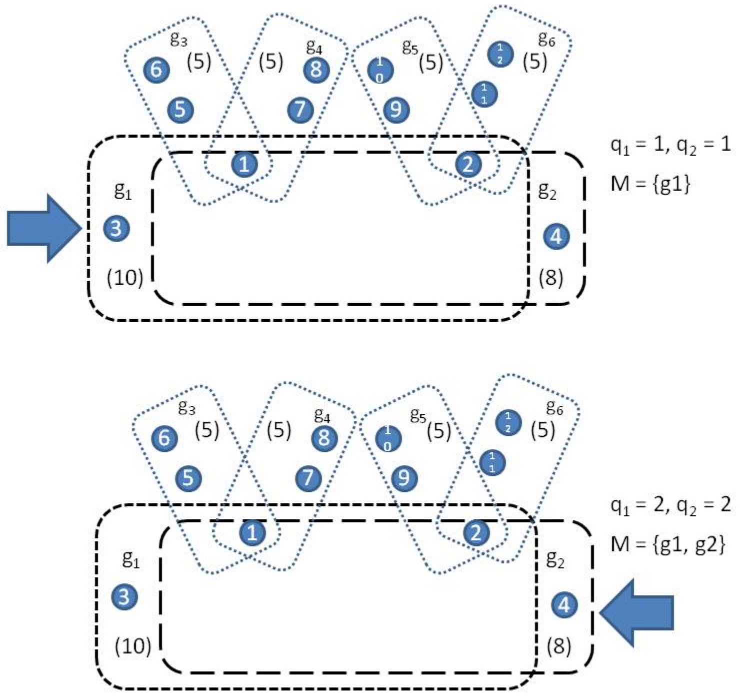


FIGURE 4.1. An example demonstrating how Algorithm 3 operates. The smaller circles represent agents, the larger circles with broken outlines represent groups. The numbers in parenthesis indicates the group's weight.

Proof: To prove that the algorithm is stable we assume that to the contrary we have a group $g \notin \mathbf{M}$ such that $w(g) > w(g')$ for $j \in g' \cap g$ and $q_j < \lambda$ where $g' \in \mathbf{M}$. Since g is not in the stable matching this means that there is an agent $i \in g$ that has already met its quota. Since the algorithm examines groups in decreasing order of weight this means that i met its quota when a group in which it is a member of g' was added to the matching. This means that $i \in g'$ such that $g' \in \mathbf{M}$ and $w(g') > w(g)$ and $q_i = \lambda$ which contradicts our initial assertion. ■

Theorem 4.1.2 *The PoA* $\leq \alpha k$ *for a simple group matching game with* $\lambda > 1$.

Proof: Let $\mathbf{M}^*$ be the optimal matching and $\mathbf{M}$ be the α-stable matching returned by Algorithm 3. Initially when the algorithm starts $\mathbf{M}^* = \mathbf{M}$. During the execution of the algorithm several groups from $\mathbf{M}^*$ are removed from $\mathbf{M}$. Let $g^* \in \mathbf{M}^* \setminus \mathbf{M}$ be one such group. There exist a $\tilde{g}$, where $\tilde{g} \in X(g^*)$ and $\tilde{g} \in \mathbf{M} \setminus \mathbf{M}^*$ such that,

$$w(g^*) < \alpha w(\tilde{g}). \tag{4.1}$$

Therefore, we can map every group $g' \in \mathbf{M}^* \setminus \mathbf{M}$ to a group $g \in \mathbf{M} \setminus \mathbf{M}^*$ such that $\alpha w(g) > w(g')$. For $g \in \mathbf{M} \setminus \mathbf{M}^*$, let $F_g \subseteq X(g)$ be the set of groups in $\mathbf{M}^* \setminus \mathbf{M}$ which map to g. Note that,

$$\bigcup_{g \in \mathbf{M} \setminus \mathbf{M}^*} F_g = \mathbf{M}^* \setminus \mathbf{M}. \tag{4.2}$$

A group $g \in \mathbf{M} \setminus \mathbf{M}^*$ can be adjacent to at most λk groups in F_g, and therefore $|F_g| \leq \lambda k$. Furthermore, a group $g' \in F_g$ can be adjacent to at most λ groups in $\mathbf{M} \setminus \mathbf{M}^*$.

From Inequality 4.1 we have,

$$\lambda k w(g) \geq w(g)|F_g| > \sum_{g' \in F_g} \frac{w(g')}{\alpha}.$$

Therefore,

$$w(g) \geq \frac{1}{\alpha k \lambda} \sum_{g' \in F_g} w(g'). \tag{4.3}$$

Each agent in g is also a member of λ groups g' each of which is a member of a set F_g, that is each $g \in \mathbf{M} \setminus \mathbf{M}^*$ can be mapped to λ sets of F_g, therefore,

$$\frac{1}{\alpha} \sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} \sum_{g' \in F_g} w(g') \geq \lambda SU(\mathbf{M}^* \setminus \mathbf{M}) \tag{4.4}$$

Thus,

$$\begin{aligned}
\sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} w(g) &\geq \frac{1}{\alpha k \lambda} \sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} \sum_{g' \in F_g} w(g') \quad \textit{using Inequality } (4.3) \\
\sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} w(g) &\geq \frac{1}{\alpha k} SU(\mathbf{M}^* \setminus \mathbf{M}) \quad \textit{using Inequality } (4.4) \\
SU(\mathbf{M} \setminus \mathbf{M}^*) &\geq \frac{1}{\alpha k} SU(\mathbf{M}^* \setminus \mathbf{M}) \qquad (4.5)
\end{aligned}$$

Since $\mathbf{M} = (\mathbf{M} \cap \mathbf{M}^*) \cup (\mathbf{M} \setminus \mathbf{M}^*)$ and $\mathbf{M} \cap \mathbf{M}^*$ is disjoint with $\mathbf{M} \setminus \mathbf{M}^*$, we get

$$\begin{aligned}
SU(\mathbf{M}) &= SU(\mathbf{M} \cap \mathbf{M}^*) + SU(\mathbf{M} \setminus \mathbf{M}^*) \\
&\geq SU(\mathbf{M} \cap \mathbf{M}^*) + \alpha k SU(\mathbf{M}^* \setminus \mathbf{M}) \quad \textit{using Inequality } (4.5) \\
&\geq \alpha k \left(SU(\mathbf{M} \cap \mathbf{M}^*) + SU(\mathbf{M}^* \setminus \mathbf{M}) \right) \\
&= k\alpha SU(\mathbf{M}^*). \qquad (4.6)
\end{aligned}$$

Consequently, we get $PoA \leq k\alpha$. ■

Theorem 4.1.3 *For any $\alpha, k \geq 1$ and $\lambda > 1$ there is a game G with maximum group size k such that the $PoA \geq \alpha(k-1)$ for α-stable matchings.*

Proof: We define a game G with $\mathcal{S} = \{g_1, g_2, \ldots, g_\lambda, g^*_{1,1}, \ldots, g^*_{k-1,\lambda}\}$ such that each agent $i \in \{1, \ldots, k-1\} \subset N$ is a member of λ groups $g^*_{i,1}, \ldots, g^*_{i,\lambda}$ and λ groups $g_1, \ldots, g_\lambda$. That is, $g_i \cap g^*_{i,j} = i$ and $g_1 \cup \cdots \cup g_{k-1} = \{1, \ldots, k-1\}$ for $i = 1, \ldots, k-1$ and $j = 1, \ldots, \lambda$ (See Figure 4.3). Every agent $j \in g_i$ has weight $c_j = \frac{1}{k-1}$ and every group $l \in (g^*_{i,j} \setminus g_i) = \frac{k\alpha - 1}{k(k-1)}$. Thus the weights of the groups are,

$$w(g_i) = \sum_{i=1}^{k} \frac{1}{k}$$

$$= 1$$

$$w(g^*_{i,j}) = \sum_{i=1}^{k-1} \frac{k\alpha - 1}{k(k-1)} + \frac{1}{k}$$

$$= \alpha$$

Let $\mathbf{M}^* = \{g^*_{i,1}, \ldots, g^*_{i,\lambda}\}$ and $\mathbf{M} = \{g_1, \ldots, g_\lambda\}$ be two matchings in the game. If all agents $i \in \{1, \ldots, k-1\}$ choose groups g_i because $u_i(\mathbf{M}) > u_i(\mathbf{M}^*)$ then the social utility in the stable matching will be,

$$SU(\mathbf{M}) = \sum_{i=1}^{\lambda} w(g_i)$$

$$= \lambda.$$

If however all agents $i \in \{1, \ldots, k-1\}$ choose groups $g^*_{i,j}$ then,

$$SU(\mathbf{M}^*) = \sum_{i=1}^{k-1} \sum_{j=1}^{\lambda} w(g^*_{i,j})$$

$$= \lambda\alpha(k-1).$$

Therefore,

$$PoS = SU(\mathbf{M}^*)/SU(\mathbf{M})$$

$$= \alpha(k-1).$$

Given a simple group matching game $G = (N, \mathcal{S})$ with $\lambda > 1$ and $\{1, \cdots, k\} \subset N$. Let $\mathbf{M}^*$ be an optimal matching and $\mathbf{M}$ be a stable matching. Given that $A = \{g_1, \cdots g_\lambda\}$ and $B_i = \{g^i_1, \cdots, g^i_\lambda\}, i = 1, \cdots, k$. Furthermore, $A \cap B_i = \{1, \cdots, k\}$. Let $A \in \mathbf{M}$ and $B_i \in \mathbf{M}^*$. Furthermore the strategies for i are $s_i = \{A, B_i\}$. The weights of the groups are as follows, $w(g) = x + \epsilon$ for all $g \in A$ and $w(g') = x$ for all $g' \in B_i, x \geq 0$.

If the agents all choose $s_i = A$ then $w(\mathbf{M}) = \lambda x$. If however the agents choose $s_i = B_i$ then $w(\mathbf{M}^*) = k\lambda x$. Therefore $PoA = w(\mathbf{M}^*)/w(\mathbf{M}) = k$.

■

4.2 Price of Stability for $\lambda > 1$

We present a description of Algorithm 4 which computes an α-stable matching with a low social utility given a game G. Every agent is given a quota of groups that it can join. Anytime an agent's group is added to $\mathbf{M}$, the current set of groups, its quota is increased. An agent's quota cannot exceed λ. The algorithm begins by setting the current set of groups $\mathbf{M}$ to $\mathbf{M}^*$ (Line 2). It then iterates through all the groups in the games and examines them in descending order of group weight. If all of the members of the currently examined group have not met their quota and the currently examined group is a factor of α greater than the groups that the agents are currently members of (Line 5) then the agents' quotas are increased (Line 8). This is repeated until all the members of the group have met their quotas (Line 5) then the groups are added to the set of groups $\mathbf{M}$ and their old groups are removed from the set (Lines 6 and 7). At the end of the execution of the algorithm the $\mathbf{M}$ return is a stable matching that is $\frac{k}{\alpha}$ times worse than the optimal matching. Furthermore, all agents who are members of the groups in $\mathbf{M}$ returned by the algorithm have met their quota, that is, they are members of not more than λ groups.

Example of Algorithm 4

Figure 4.2 shows an example of Algorithm 4 in operation. The algorithm begins with $\mathbf{M}$ being the optimal matching, it then examines the groups in decreasing order of weight. In this example $k = 3$, $\alpha = 2$ and $\lambda = 2$.

In the first iteration the algorithm examines group g_1. Since its members, 1, 2 and 3 have not met their quotas and its weight is a factor of α greater than its neighbors g_3, g_4, g_5 and g_6 in $\mathbf{M}$, its members quotas are increased and it is added to a temporary matching T.

Algorithm 4: Algorithm to find a stable matching in a simple group packing game with $\lambda > 1$

```
input  : A group matching game $G = \langle N, \mathcal{S}, (c_i), (S_i), (u_i) \rangle$
output: A set of groups $\mathbf{M}$
1 $Z = (g_1, g_2, \ldots, g_{|\mathcal{S}|})$ is a sorted list of groups were $w(g_m) > w(g_n), m > n, g_m, g_n \in \mathcal{S}$;
2 $\mathbf{M} \leftarrow \mathbf{M}^*$;
3 $q_i \leftarrow 1$ for all $i \in N$ ; // each agent $i$ has a quota $q_i$. We initialize the quotas to 1
4 for $j = 1, \ldots, |\mathcal{S}|$ do
5     if no $i \in g_j$ have exceeded their quota and $\frac{w(g_j)}{\alpha} > w(g')$ for all $g' \in X(g) \wedge g' \in \mathbf{M}$ then
6         $\mathbf{M} \leftarrow \mathbf{M} \cup g_j$;
7         $\mathbf{M} \leftarrow \mathbf{M} \setminus g'$;
8         $q_i \leftarrow q_i + 1$ for all $i \in q_i$;
9 return $\mathbf{M}$;
```

In the next iteration g_2 is examined. Its weight is a factor of α greater than its neighbors g_3, g_4, g_5 and g_6. In addition its members 1 and 2 have now met their quotas therefore g_1 is taken from T and together with g_2, is added to $\mathbf{M}$ and its neighbors g_3, g_4, g_5 and g_6 are removed from $\mathbf{M}$.

Note that the final matching, comprising of g_1 and g_2, is 2-stable.

Theorem 4.2.1 *Given a game G, Algorithm 4 returns a stable matching.*

Proof: Let $\mathbf{M}$ be the output of then algorithm. In $\mathbf{M}$ an agent is added to at most one group. This is because when its group is added to $\mathbf{M}$ all other groups that it belongs to are not eligible to be added to $\mathbf{M}$ in subsequent iterations. Therefore, $\mathbf{M}$ is a matching.

Next we will prove the $\mathbf{M}$ is stable. Suppose to the contrary that $\mathbf{M}$ is unstable. This means that there is a group $g \notin \mathbf{M}$ such that $\frac{w(g)}{\alpha} > w(g')$ for $j \in g' \cap g$, $g' \in X(g)$, and $q_j < \lambda$ where $g' \in \mathbf{M}$. Since g' is in $\mathbf{M}$ then from line **??** we have it that, $\frac{w(g')}{\alpha} > w(g'')$ for all $g'' \in X(g')$. This is a contradiction because $g \in X(g')$ but $\frac{w(g)}{\alpha} > w(g')$. ■

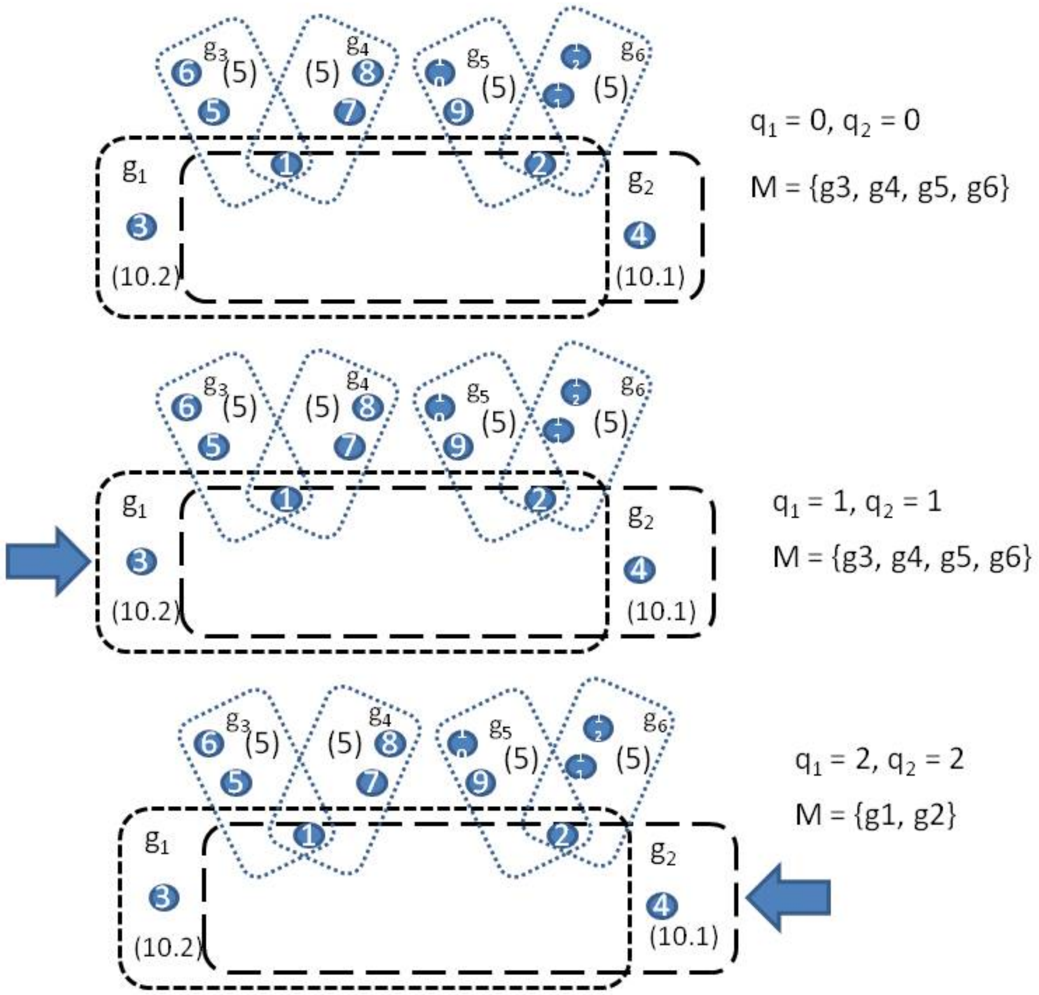


FIGURE 4.2. An example demonstrating how Algorithm 4 operates. The smaller circles represent agents, the larger circles with broken outlines represent groups. The numbers in parenthesis indicates the group's weight.

Theorem 4.2.2 *The PoS of an α-stable simple group matching game with $\lambda > 1$ is at most k/α.*

Proof: Let $\mathbf{M}^*$ be the optimal matching and $\mathbf{M}$ be the α-stable matching returned by Algorithm 4. Initially when the algorithm starts $\mathbf{M}^* = \mathbf{M}$. During the execution of the algorithm several groups from $\mathbf{M}^*$ are removed from $\mathbf{M}$. Let $g^* \in \mathbf{M}^* \setminus \mathbf{M}$ be one such group. There exist a $\tilde{g}$, where $\tilde{g} \in X(g^*)$ and $\tilde{g} \in \mathbf{M} \setminus \mathbf{M}^*$ such that,

$$\alpha w(g^*) < w(\tilde{g}). \tag{4.7}$$

Therefore, we can map every group $g' \in \mathbf{M}^* \setminus \mathbf{M}$ to a group $g \in \mathbf{M} \setminus \mathbf{M}^*$ such that $w(g) > \alpha w(g')$. For $g \in \mathbf{M} \setminus \mathbf{M}^*$, let $F_g \subseteq X(g)$ be the set of groups in $\mathbf{M}^* \setminus \mathbf{M}$ which map to g. Note that,

$$\bigcup_{g \in \mathbf{M} \setminus \mathbf{M}^*} F_g = \mathbf{M}^* \setminus \mathbf{M}. \tag{4.8}$$

A group $g \in \mathbf{M} \setminus \mathbf{M}^*$ can be adjacent to at most λk groups in F_g, and therefore $|F_g| \leq \lambda k$. Furthermore, a group $g' \in F_g$ can be adjacent to at most λ groups in $\mathbf{M} \setminus \mathbf{M}^*$.

From Inequality 4.7 we have,

$$\lambda k w(g) \geq w(g)|F_g| > \sum_{g' \in F_g} \alpha w(g').$$

Therefore,

$$w(g) \geq \frac{\alpha}{k\lambda} \sum_{g' \in F_g} w(g'). \tag{4.9}$$

Each agent in g is also a member of λ groups g' each of which is a member of a set F_g, that is each $g \in \mathbf{M} \setminus \mathbf{M}^*$ can be mapped to λ sets of F_g, therefore,

$$\alpha \sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} \sum_{g' \in F_g} w(g') \geq \lambda SU(\mathbf{M}^* \setminus \mathbf{M}) \tag{4.10}$$

Thus,

$$\sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} w(g) \geq \frac{\alpha}{k\lambda} \sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} \sum_{g' \in F_g} w(g') \quad \textit{using Inequality } (4.9)$$

$$\sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} w(g) \geq \frac{\alpha}{k} SU(\mathbf{M}^* \setminus \mathbf{M}) \quad \textit{using Inequality } (4.10)$$

$$SU(\mathbf{M} \setminus \mathbf{M}^*) \geq \frac{\alpha}{k} SU(\mathbf{M}^* \setminus \mathbf{M}) \tag{4.11}$$

Since $\mathbf{M} = (\mathbf{M} \cap \mathbf{M}^*) \cup (\mathbf{M} \setminus \mathbf{M}^*)$ and $\mathbf{M} \cap \mathbf{M}^*$ is disjoint with $\mathbf{M} \setminus \mathbf{M}^*$, we get

$$\begin{aligned} SU(\mathbf{M}) &= SU(\mathbf{M} \cap \mathbf{M}^*) + SU(\mathbf{M} \setminus \mathbf{M}^*) \\ &\geq SU(\mathbf{M} \cap \mathbf{M}^*) + \frac{\alpha}{k} SU(\mathbf{M}^* \setminus \mathbf{M}) \quad \textit{using Inequality } (4.11) \\ &\geq \frac{\alpha}{k} \left(SU(\mathbf{M} \cap \mathbf{M}^*) + SU(\mathbf{M}^* \setminus \mathbf{M}) \right) \\ &= \frac{\alpha}{k} SU(\mathbf{M}^*). \end{aligned} \tag{4.12}$$

Consequently, we get $PoS \leq \frac{k}{\alpha}$. ■

Theorem 4.2.3 *There is a game G with an α-stable matching and $\lambda \geq 1$ such that $PoS \geq (k-1)/\alpha$ for $k \geq \alpha$.*

Proof: We define a game G with $\mathcal{S} = \{g_1, g_2, \ldots, g_\lambda, g^*_{1,1}, \ldots, g^*_{k-1,\lambda}\}$ such that each agent $i \in \{1, \ldots, k-1\} \subset N$ is a member of λ groups $g^*_{i,1}, \ldots, g^*_{i,\lambda}$ and λ groups $g_1, \ldots, g_\lambda$. That is, $g_i \cap g^*_{i,j} = i$ and $g_1 \cup \cdots \cup g_{k-1} = \{1, \ldots, k-1\}$ for $i = 1, \ldots, k-1$ and $j = 1, \ldots, \lambda$ (See Figure 4.3). Every agent $j \in g_i$ has weight $c_j = \frac{1}{k-1}$ and every group $l \in (g^*_{i,j} \setminus g_i) = \frac{k-\alpha}{k\alpha(k-1)}$. Thus the weights of the groups are,

$$w(g_i) = \sum_{i=1}^{k} \frac{1}{k}$$
$$= 1$$
$$w(g^*_{i,j}) = \sum_{i=1}^{k-1} \frac{k-\alpha}{k\alpha(k-1)} + \frac{1}{k}$$
$$= \frac{1}{\alpha}$$

Let $\mathbf{M}^* = \{g^*_{i,1}, \dots, g^*_{i,\lambda}\}$ and $\mathbf{M} = \{g_1, \dots, g_\lambda\}$ be two matchings in the game. If all agents $i \in \{1, \dots, k-1\}$ choose groups g_i because $u_i(\mathbf{M}) > u_i(\mathbf{M}^*)$ then the social utility in the stable matching will be,

$$SU(\mathbf{M}) = \sum_{i=1}^{\lambda} w(g_i)$$
$$= \lambda.$$

If however all agents $i \in \{1, \dots, k-1\}$ choose groups $g^*_{i,j}$ then,

$$SU(\mathbf{M}^*) = \sum_{i=1}^{k-1} \sum_{j=1}^{\lambda} w(g^*_{i,j})$$
$$= \frac{\lambda(k-1)}{\alpha}.$$

Therefore,

$$PoS = SU(\mathbf{M}^*)/SU(\mathbf{M})$$
$$= \frac{k-1}{\alpha}$$

■

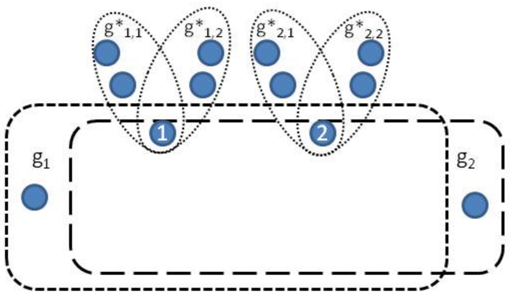


FIGURE 4.3. An example of a group matching game with $\lambda = 2$ and $k = 3$. The small circles represent agents and the larger circles with broken outlines represent groups. Agent $1 \in g_1, g_2, g^*_{1,1}, g^*_{1,2}$ and agent $2 \in g_1, g_2, g^*_{2,1}, g^*_{2,2}$. Groups g_1, g_2 form the stable matching. Groups $g^*_{1,1}, g^*_{1,2}, g^*_{2,1}, g^*_{2,2}$ form the optimal matching.

Chapter 5
Group Matching Game with Subset Property

In this section we introduce games that exhibit a so-called *subset property*. These games are like the games we described earlier with an additional caveat that the subset of every group in $\mathcal{S}$ is also a valid strategy.

Formally, we define these games as follows. Given a game $G = \langle N, \mathcal{S}, (c_i), (S_i), (u_i) \rangle$ such that for an agent i, $g \in S_i \Rightarrow 2^g \in S_i$. That is, if an agent has g as one of its strategies then all subsets of g are also strategies of that agent. In addition, all the agents have the same weight therefore we can set $c_i = 1$, $w(g) = |g|$ and $SU(\mathbf{M}) = \sum_{g \in \mathbf{M}} w(g)$. We can then prove the following theorem.

5.1 PoA Analysis

Theorem 5.1.1 *The $PoA \leq 2$ for a game G in which all agent have the same weight.*

Proof: Let $\mathbf{M}^*$ be the optimal matching and $\mathbf{M}$ be an α-stable matching. We proceed by setting an upper-bound for $SU(\mathbf{M}^*)$ with respect to $SU(\mathbf{M})$ which will give the PoA bound. Let $g^* \in \mathbf{M}^* \setminus \mathbf{M}$, then there exist a $\tilde{g}$, where $\tilde{g} \in X(g^*)$ and $\tilde{g} \in \mathbf{M} \setminus \mathbf{M}^*$, such that,

$$w(\tilde{g}) \geq w(g^*), \tag{5.1}$$

since $\mathbf{M}$ is an stable matching, at least one agent $i \in g \cap g^*$ gets a utility that is better in $\mathbf{M}$ than in $\mathbf{M}^*$.

Furthermore, we can say that $SU(\mathbf{M}^*) > SU(\mathbf{M})$ if there exist an agent $i \in g^*$, such that $|g^*| = 2$ and $S_i = \{g^*\}$. That is, there will be a loss in social utility if an agent in a group looses its only partner and cannot form another group. Therefore in this case we have,

$$w(g^*) = 2, \tag{5.2}$$

We can therefore map every group $g' \in \mathbf{M}^* \setminus \mathbf{M}$ to a group $g \in \mathbf{M} \setminus \mathbf{M}^*$ such that $w(g') \leq w(g)$. For $g \in \mathbf{M} \setminus \mathbf{M}^*$, let $F_g \subseteq X(g)$ be the set of groups in $\mathbf{M}^* \setminus \mathbf{M}$ which map to g. Note that,

$$\bigcup_{g \in \mathbf{M} \setminus \mathbf{M}^*} F_g = \mathbf{M}^* \setminus \mathbf{M}. \tag{5.3}$$

A group $g \in \mathbf{M} \setminus \mathbf{M}^*$ can be adjacent to at most k groups in F_g, and therefore $|F_g| \leq k$. From Inequality 5.1 we have,

$$kw(g) \geq w(g)|F_g| \geq \sum_{g' \in F_g} w(g').$$

Therefore,

$$w(g) \geq \frac{1}{k} \sum_{g' \in F_g} w(g'). \tag{5.4}$$

Thus,

$$\begin{aligned}
SU(\mathbf{M} \setminus \mathbf{M}^*) &= \sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} w(g) \\
&\geq \frac{1}{k} \sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} \sum_{g' \in F_g} w(g') \quad \textit{using Inequality } (5.4) \\
&\geq \frac{1}{k} \sum_{g \in \mathbf{M} \setminus \mathbf{M}^*} \sum_{g' \in F_g} 2 \quad \textit{using Inequality } (5.4) \textit{ and} (5.2) \\
&\geq \frac{1}{k} \sum_{g' \in \mathbf{M}^* \setminus \mathbf{M}} 2k \quad \textit{using Inequality } (5.3) \\
&= 2 \cdot SU(\mathbf{M}^* \setminus \mathbf{M}).
\end{aligned} \tag{5.5}$$

Since $\mathbf{M} = (\mathbf{M} \cap \mathbf{M}^*) \cup (\mathbf{M} \setminus \mathbf{M}^*)$ and $\mathbf{M} \cap \mathbf{M}^*$ is disjoint with $\mathbf{M} \setminus \mathbf{M}^*$, we get

$$\begin{aligned}
SU(\mathbf{M}) &= SU(\mathbf{M} \cap \mathbf{M}^*) + SU(\mathbf{M} \setminus \mathbf{M}^*) \\
&\geq SU(\mathbf{M} \cap \mathbf{M}^*) + 2SU(\mathbf{M}^* \setminus \mathbf{M}) \quad \textit{using Inequality } (5.5) \\
&\geq 2\left(SU(\mathbf{M} \cap \mathbf{M}^*) + SU(\mathbf{M}^* \setminus \mathbf{M})\right) \\
&= 2 \cdot SU(\mathbf{M}^*).
\end{aligned} \tag{5.6}$$

Since $\mathbf{M}$ is an arbitrary stable matching, we get $PoA \leq 2$. ■

5.2 Lower Bound

Theorem 5.2.1 *There is a game G where all the agents have the same weight and with maximum group size k such that the $PoA \geq 2$ for stable matchings.*

Proof: We define a game G with $\mathcal{S} = \{g, g_1, \ldots, g_k\}$ such that $g_i \cap g_j = \emptyset$ for $i \neq j$ and g shares exactly one agent with each g_i, that is $|g \cap g_i| = 1$ (see Figure 3.2). Therefore, the group g is adjacent to k other groups. Furthermore, $|g| = k$ and $|g_i| = 2$. Every agent $i \in N$ has weight $c_i = 1$. Thus the weights of the groups are,

$$w(g) = \sum_{i=1}^{k} 1 = k$$

and

$$w(g_i) = 2.$$

The optimal matching is $\mathbf{M}^* = \{g_1, \ldots, g_k\}$ with $SU(\mathbf{M}^*) = \sum_{i=1}^{k} w(g_i) = 2k$. The matching $\mathbf{M} = \{g\}$ is stable since each agent $p \in g_i \cap g$ will choose g because it improves it utility by moving to g_i, since $u_p(\mathbf{M}^*) \leq u_p(\mathbf{M})$. The only other stable matching is $\mathbf{M}^*$. Therefore $SU(\mathbf{M}) = k$ and

$$\begin{aligned} PoA &= SU(\mathbf{M}^*)/SU(\mathbf{M}) \\ &= 2. \end{aligned}$$

■

Chapter 6
Group Cover Game

In this section we introduce group cover games. A group cover game $G = \langle N, \mathcal{S}, (c_i), (S_i), (u_i) \rangle$. These games differ from the group matching games in that $SU(\mathbf{M}) = 1/|\mathbf{M}|$.

6.1 Price of Anarchy for Group Cover Game

Theorem 6.1.1 *The PoA of an anonymous, agent weighted game with* $SU(\mathbf{M}) = 1/|\mathbf{M}|$ *is* $log_e n$.

Proof: Let G be group cover game. Suppose that the number of groups in the optimal matching $\mathbf{M}^*$ is m, i.e. $SU(\mathbf{M}^*) = 1/m$. We examine the groups in descending order of weight. Let n_1 be the number of agents left after all the agents in the highest weighted group have been removed and n_2 is the number of agents left after all the first and second highest weighted agents have been removed etc. We can say that the first group must have size not less that n/m agents (If it had group size less than that then its members would defect to the optimal matching). So therefore, after the agents in the first group are matched we can say that the number of agents remaining is

$$n_1 \leq n - n/m = n(1 - 1/m)$$

. After all the agents in the first group have been removed we know that at least one group will have no less than n_1/m agents (the optimal matching now has n_1 remaining agents and m groups), therefore we are left with $n_2 \leq n_1 - n_1/m \leq n(1 - 1/m)n^2$ agents. We can see that generally,

$$n_{i+1} \leq n(1 - 1/m)n^i$$

. To determine the number of groups needed for all the agent to be matched we proceed as follows,

$$\begin{aligned}
n(1-1/m)^k &< 1\\
n(1-1/m)^{mk/m} &< 1\\
(1-1/m)^{mk/m} &< 1/n\\
e^{k/m} &< 1/n \; \ldots \mathit{based\ on\ the\ relation}\ (1-1/x)^{\frac{1}{x}} \approx e^{-1}\\
k &< m\log_e n
\end{aligned}$$

We can see that, $SU(\mathbf{M}^*) = \frac{1}{m}$ and $SU(\mathbf{M}) = \frac{1}{m\log_e n}$. Therefore, $PoA = \log_e n$. ■

Chapter 7
Conclusions to Group Matching Games

Several contributions have been made by the work outlined in this dissertation. We show that group matching games have several real world applications in areas such as kidney exchange programs, social networks and wireless sensor networks. We have described new types of matching games and have put forward a game theoretic framework for describing and analyzing these games. We then presented several intuitive alogithms for achieving stablility in these games. We have also outlined the performance of these games in terms of the PoA and PoS matric. Specifically we discovered that the PoA is bounded by the maximum group size. By relaxing the criteria for stability the PoA increases by a factor of α and the PoS improves by a factor of α. We also discovered, counterintuitively, that by allowing agents to join up to λ groups we do not improve on the PoA and PoS bounds significantly.

These results can serve as a basis to use to further explore more elaborate matching games in a game theoretic setting. Future work could involve exploring more complex games such as games in which the utility that an agent gains from a group is its Shapely value [4] which depends on its contribution to the group. We could also investigate games in which agents act altruistically. That is, agent are willing to loose a certain amount of utility in order to help group members to improve their own utilities. Using group matching games as a basis we can explore whether allowing this will improve the PoA and PoS.

Part II

Bottleneck Routing Games on Grids

Chapter 8
Introduction to Bottleneck Routing Games on Grids

Motivated by the selfish behavior of entities in communication networks and tasks in job scheduling, we study congestion games where each packet's path is controlled independently by a selfish player. We consider non-cooperative routing games with n players, where each player's *pure strategy set* consists of a set of paths in the network. A player selfishly selects a strategy (a single path) that maximizes the player's utility cost function. Such games are also known as *atomic* or *unsplittable-flow* games. We focus on *bottleneck routing games* where the objective for the social outcome is to minimize the bottleneck congestion C, the maximum congestion on any edge. Each player's objective is also to select a path with the smallest bottleneck congestion along the selected path's edges. Typically, the congestion on a edge is a non-decreasing function on the number of paths that use the edge; here, we consider the congestion to be simply the number of players that use the edge.

Bottleneck routing games have been studied in the literature [7, 9, 8]. In [7] the authors observe that bottleneck games are important in networks for various practical reasons. In wireless networks, the maximum congested link is related to the lifetime of the network since the nodes adjacent to high congestion links transmit large number of packets which results to higher energy depletion. High congestion links also result to congestion hot-spots which may slow-down the network throughput. Hot spots also increase the vulnerability of the network to malicious attacks which aim to increase the congestion of links in the hope to bring down the network. Bottleneck games are also important from a theoretical point of view since the bottleneck congestion is immediately related to optimal packet scheduling. In a seminal result, Leighton *et al.* [21] showed that there exist packet scheduling algorithms that deliver the packets along their chosen paths in time very close to $C + D$, where D is the maximum chosen path length. When $C \gg D$, the congestion becomes the dominant factor in the packet scheduling performance. Thus, smaller bottleneck congestion C immediately implies faster packet delivery time.

A natural problem that arises in games concerns the effect of the players' selfishness on the welfare of the whole system measured with the *social cost* C. We examine the consequence of the selfish behavior in pure *Nash equilibria* which are stable states of the game in which no player can unilaterally improve her situation. We quantify the effect of selfishness with the *price of anarchy* (PoA) [19, 25], which expresses how much larger is the worst social cost in a Nash equilibrium compared to the social cost in the optimal coordinated solution in the strategy space. The price of anarchy provides a measure for estimating how closely do Nash equilibria of bottleneck congestion games approximate the optimal C^* of the respective coordinated optimization problem in the player's strategy set.

Ideally, the price of anarchy should be small. However, the current literature results have only provided weak bounds for bottleneck games. In [7] it is shown that if the resource congestion delay function is bounded by some polynomial with degree k then $PoA = O(|E|^k)$, where E is the set of edges in the graph. In [9] it is shown that if $k = 1$ there are game instances with $PoA = \Omega(|E|)$. A natural question that we explore here is the circumstances under which there are bottleneck games with alternative and better price of anarchy bounds.

8.1 Contributions

We consider grid network topologies in which the nodes are placed in a d-dimensional array and each node connects with edges to at most $2d$ neighbors. The number of nodes is $n^d = N$. Grid networks have been used as interconnection networks in parallel multiprocessor computer architectures [22]. In wireless networks 2-dimensional grids provide a framework for formulating and analyzing wireless communication problems. In other communication networks routing and scheduling algorithms are typically first tested and analyzed on grids and then extended to arbitrary network topologies [10].

We explore games where the price of anarchy is expressed in terms of the numbers of bends that the paths use in the grid. A *bend* is a node in a path which connects two path segments in different dimensions. We explore games where the strategies of the players consists of paths whose bends are bounded by β, where β can be any number of nodes up to N. We first examine basic bottleneck games on grids with at most β bends for the paths. We show that there are instances in the 2-dimensional grid with $\beta = O(1)$ and price of anarchy $\Omega(\sqrt{N})$. However, this is not satisfactory. In order to obtain price of anarchy bounded by β, we explore two alternative games.

In the first game we utilize *channels*, where path segments on straight lines are routed in different channels according to their lengths. An edge accommodates $\alpha = \log n$ channels (logarithms are base 2), such that channel j is used by path segments of length in range $[2^j, 2^{j+1} - 1]$. Channels do not interfere with each other so that congestion can be created only by path segments in the same channel. Channels can be implemented with different frequencies in the physical communication medium, or with time division multiplexing, or with other means of signal multiplexing. The use of channels enables us to control the price of anarchy. We show that in *channel bottleneck games* if paths are allowed to use at most β bends, the price of anarchy is

$$PoA = O((\beta/d) \log N).$$

We also provide a lower bound $PoA = \Omega(\beta)$. Thus, for constant d, the upper bound is tight within a $\log n$ factor.

We then explore games which use only one channel. Now, in order to control the price of anarchy we split the path segments into different grid lines according to the lengths of the segments. Odd lines with index $2i + 1$ are used to route path segments of length in range $[2^{i \bmod \alpha}, 2^{(i \bmod \alpha)+1} - 1]$, where $\alpha = \log n$ (logarithms are base 2). Even index lines are used to route paths segments with length at most $2\alpha - 1$. Even index lines are uses to route paths close to the source and destination and when path segments switch to different lengths. This gives $\alpha + 1$ different types of lines. Thus, path segments are separated in space, and a single channel suffices. Note that we can still perform routing from every node to any other node without significantly increasing the number of bends, compared to a routing mode without space separated path segments. We show that in the respective *split bottleneck games* if paths are allowed to use at most β bends, the price of anarchy is

$$PoA = O((\beta/d^2)\log^2 N).$$

We also provide a lower bound $PoA = \Omega(\beta)$. Thus, for constant d, the upper bound is tight within a $\log^2 n$ factor.

8.2 Impact of Games with Small Number of Bends

We demonstrate that Nash equilibria of bottleneck games with small number of bends can approximate efficiently the best coordinated solution that uses an arbitrary number of bends. Assuming that every path in the network can be used, there exist oblivious routing algorithms in grids which find paths with $O(d \log N)$ bends and achieve $O(d \log N)$ approximation to the optimal solution that uses an arbitrary number of bends [10]. Let $\mathbf{C}$ denote the solution returned by the oblivious algorithm and $\mathbf{C}^*$ denote the global optimal solution with an arbitrary number of bends. Clearly, $\mathbf{C}/\mathbf{C}^* = O(d \log N)$.

Consider now channel bottleneck games where the strategy of each player contains all possible paths in the grid with $\beta = O(d \log N)$ bends. Let C^* denote the smallest social cost. Clearly, $C^* \leq \mathbf{C}$. Let C be any Nash equilibrium of the game. Since $C/C^* \leq PoA$, and $PoA = O((\beta/d)\log N) = O(\log^2 N)$, we obtain

$$C/\mathbf{C}^* = O(d \log^3 N).$$

Therefore, Nash equilibria of channel bottleneck games with small number of bends provide good approximations to the optimal coordinated solution with arbitrary number of bends.

We can obtain a similar result for split bottleneck games. Note that any solution of an oblivious routing algorithm with congestion C' and x bends is translated to a solution with congestion $C' \cdot \log n$ and $O(x)$ bends in the split grid, since some of the path segments have to be rerouted to nearby lines that accommodate their length. Since $PoA = O((\beta/d^2)\log^2 N) = O((1/d) \cdot \log^3 N)$, we obtain:

$$C/\mathbf{C}^* = O((1/d)\log^5 N).$$

8.3 Related Work

Congestion games were introduced and studied in [24, 26]. In [26], Rosenthal proves that congestion games have always pure Nash equilibria. Koutsoupias and Papadimitriou [19] introduced the notion of price of anarchy in the specific *parallel link networks* model in which they provide the bound $PoA = 3/2$. Roughgarden and Tardos [29] provided the first result for splittable flows in general networks in which they showed that $PoA \leq 4/3$ for a player cost which reflects to the sum of congestions of the resources of a path. Pure equilibria with atomic flow have been studied in [9, 12, 23, 34] (our work fits into this category), and with splittable flow in [27, 28, 29, 30]. Most of the work in the literature uses a player cost metric related to the aggregate sum of congestions on all the edges of the player's path; and the social cost metric is also an aggregate expression of all the edge congestions [12, 28, 29, 30, 34].

Bottleneck routing games have been studied in [7], where the authors consider the maximum congestion metric in general networks with splittable and atomic flow. They prove the existence and non-uniqueness of equilibria in both the splittable and atomic flow models. They show that finding the best Nash equilibrium that minimizes the social cost is a NP-hard problem. Further, they show that the price of anarchy may be unbounded for specific resource congestion functions. In [9], the authors consider bottleneck routing games in general networks where they prove that $\ell \leq PoA \leq c(\ell^2 + log^2|V|)$, where ℓ is the size of the largest edge-simple cycle in the graph and c is a constant. In [8] the authors consider bottleneck games with the $C + D$ metric. In [15], the authors prove the existence of strong Nash equilibria (which concern coalitions of players) for games with the lexicographic improvement property; such games include the bottleneck routing games that we consider here. In [18], the authors provide games with the bottleneck social cost which achieve low price of anarchy when the players use a cost function which is an aggregate exponential expression of the congestions of the edges in their selected paths.

8.4 Outline

In Chapter 9 we give basic definitions. In Chapter 10 we present a basic bottleneck routing game with high price of anarchy. In Section 11.1 we present the channel and split bottleneck games, respectively, which achieve price of anarchy bounded by the number of bends β. We finish with providing lower bounds in Section 12.2.

Chapter 9
Definitions for Bottleneck Routing Games

The *d-dimensional grid* $G = (V, E)$ consists $N = |V| = n^d$ nodes arranged in a grid of d dimensions with side length n in each dimension. There is an edge connecting a node with each of its $2d$ neighbors (except for the nodes at the boundaries of the grid). Each node has a coordinate $(a_1, a_2, \ldots, a_d)$, where $a_i \in [0, n-1]$ denotes the position in the ith dimension. An example of a 2-dimensional grid is shown in Figure 10.1. A line segment with x edges in the kth dimension is a sequence of nodes $(a_1, \ldots, a_k, \ldots, a_d), \ldots, (a_1, \ldots, a_k + x, \ldots, a_d)$.

Let $\Pi = \{\pi_1, \ldots, \pi_\kappa\}$ be a set of players such that each π_i corresponds to a path request from a source u_i and destination v_i. A *routing* $\mathbf{p} = [p_1, p_2, \cdots, p_\kappa]$ is a collection of paths, where p_i is a path for player π_i from u_i to v_i. For any routing $\mathbf{p}$ and any edge $e \in E$, the *edge-congestion* $C_e(\mathbf{p})$ is the number of paths in $\mathbf{p}$ that use edge e. For any path q, the *path-congestion* $C_q(\mathbf{p})$ is the maximum edge congestion over all edges in q, namely, $C_q(\mathbf{p}) = \max_{e \in q} C_e(\mathbf{p})$. Player's π_i congestion is denoted as $C_{\pi_i}(\mathbf{p}) = C_{p_i}(\mathbf{p})$. The *network (bottleneck) congestion* $C(\mathbf{p})$ is the maximum edge-congestion over all edges in E, that is, $C(\mathbf{p}) = \max_{e \in E} C_e(\mathbf{p})$.

We denote the length (number of edges) of any path p as $|p|$. For a grid G, the path p consists of a sequence path segments which change dimensions. A *bend* of a path is a node that connects two consecutive path segments in different dimensions. By default, we take the source and destination nodes to be bends.

A routing game in graph G is a tuple $\mathbf{R} = (G, \Pi, \mathcal{P})$, where $\Pi = \{\pi_1, \pi_2, \ldots, \pi_\kappa\}$ is the set of players such that each player π_i has a source node u_i and destination v_i. The set $\mathcal{P}$ is the strategy state space of the game, $\mathcal{P} = \mathcal{P}_1 \times \mathcal{P}_2 \times \cdots \times \mathcal{P}_\kappa$, where $\mathcal{P}_i$ is the *strategy set* of player π_i which is a collection of available paths in G for player i from u_i to v_i. Any path

$p \in \mathcal{P}_i$ is a *pure strategy* available to player π_i. A *pure strategy profile (or game state)* is any routing $\mathbf{p} = [p_1, p_2, \cdots, p_\kappa] \in \mathcal{P}$.

For game $\mathbf{R}$ and routing $\mathbf{p}$, the *social cost* (or *global cost*) is a function of routing $\mathbf{p}$, and it is denoted $SC(\mathbf{p})$. The *player or local cost* is also a function on $\mathbf{p}$ denoted $pc_i(\mathbf{p})$. We use the standard notation $\mathbf{p}_{-i}$ to refer to the collection of paths $\{p_1, \cdots, p_{i-1}, p_{i+1}, \cdots, p_\kappa\}$, and $(p_i; \mathbf{p}_{-i})$ as an alternative notation for $\mathbf{p}$ which emphasizes the dependence on p_i. A *greedy move* is available to player π_i if the player can obtain lower cost by changing the current path from p_i to p'_i. Specifically, the greedy move takes the original routing $\mathbf{p} = (p_i; p_{-i})$ to $\mathbf{p}' = (p'_i; p_{-i})$ (in which path p_i is replaced by p'_i) such that $pc_i(\mathbf{p}') < pc_i(\mathbf{p})$.

Player i is *locally optimal* (or *stable*) in routing $\mathbf{p}$ if $pc_i(\mathbf{p}) \leq pc_i(p'_i; \mathbf{p}_{-i})$ for all paths $p'_i \in \mathcal{P}_i$. In other words, no greedy move is available for a locally optimal player. A routing $\mathbf{p}$ is in a Nash Equilibrium if every player is locally optimal. Nash Equilibria quantify the notion of a stable selfish outcome. A routing $\mathbf{p}^* \in \mathcal{P}$ is an *optimal pure strategy profile* if it has minimum attainable social cost: for any other pure strategy profile $\mathbf{p} \in \mathcal{P}$, $SC(\mathbf{p}^*) \leq SC(\mathbf{p})$.

We quantify the quality of the Nash Equilibria with the *price of anarchy* (PoA) (sometimes referred to as the coordination ratio) and the *price of stability* (PoS). Let $\mathbf{Q}$ denote the set of distinct Nash Equilibria, and let SC^* denote the social cost of the optimal routing $\mathbf{p}^* \in \mathcal{P}$. Then,

$$PoA = \sup_{\mathbf{p} \in \mathbf{Q}} \frac{SC(\mathbf{p})}{SC^*}, \qquad PoS = \inf_{\mathbf{p} \in \mathbf{Q}} \frac{SC(\mathbf{p})}{SC^*}.$$

Chapter 10
Basic Bottleneck Routing Game

Consider a routing game $\mathbf{R} = (G, \Pi, \mathcal{P})$ in a d-dimensional grid $G = (V, E)$, where each path in $\mathcal{P}_i$ is allowed to have at most β bends. For any routing $\mathbf{p} = [p_1, p_2, \cdots, p_\kappa] \in \mathcal{P}$, the social cost function is the bottleneck congestion, $SC(\mathbf{p}) = C(\mathbf{p})$, and the player cost function is the bottleneck congestion of its path, $pc_i(\mathbf{p}) = C_{\pi_i}(\mathbf{p}) = C_{p_i}(\mathbf{p})$.

We first show that such (basic) games have always Nash equilibria and the price of stability is 1. However, there are game instances where the price of anarchy is very large compared to the number of bends β. For this reason we explore alternative games with low price of anarchy in Sections 11.1 and 12.1.

The stability of the above basic game follows from techniques in [9, 15] related to potential functions based on lexicographic ordering. We give the details here for completeness. For routing $\mathbf{p}$, the *congestion vector*

$$M(\mathbf{p}) = [m_0(\mathbf{p}), m_1(\mathbf{p}), \ldots, m_\kappa(\mathbf{p})]$$

is defined such that each component $m_j(\mathbf{p})$ is the number of edges with congestion j. Note that

$$\sum_j m_j(\mathbf{p}) = |E|.$$

The network congestion $C(\mathbf{p})$ is the maximum index j for which $m_j > 0$. We define a lexicographic total order on routings according to their congestion vectors. Let $\mathbf{p}$ and $\mathbf{p}'$ be two routings, with $M(\mathbf{p}) = [m_0, m_1, \ldots, m_\kappa(\mathbf{p})]$, and $M(\mathbf{p}') = [m'_0, m'_1, \ldots, m'_\kappa(\mathbf{p})]$. Two routings are equal, written $\mathbf{p} = \mathbf{p}'$, if and only if $m_j = m'_j$ for all $j \geq 0$. Routing $\mathbf{p}$ is smaller than $\mathbf{p}'$, written $p < p'$, if and only if there is some $j \in [0, \kappa]$ such that $m_j < m'_j$ and $\forall j' > j, m_{j'} = m'_{j'}$.

It is easy to verify that for any greedy move of a player from a routing $\mathbf{p}$ to routing $\mathbf{p}'$ it holds that $\mathbf{p}' < \mathbf{p}$, since a lower index vector position increases in $M(\mathbf{p}')$ and a higher index vector position decreases in $M(\mathbf{p}')$ with respect to $M(\mathbf{p})$. Let $\mathbf{p}^* \in \mathcal{P}$ be the minimum routing (according to the total lexicographic order) in the available game state set. Routing $\mathbf{p}^*$ is a Nash equilibrium since no player can perform a greedy move to improve its cost. Further, $\mathbf{p}^*$ has optimal social cost, since if there was another state with smaller social cost then $\mathbf{p}^*$ wouldn't be minimum. Therefore, we obtain:

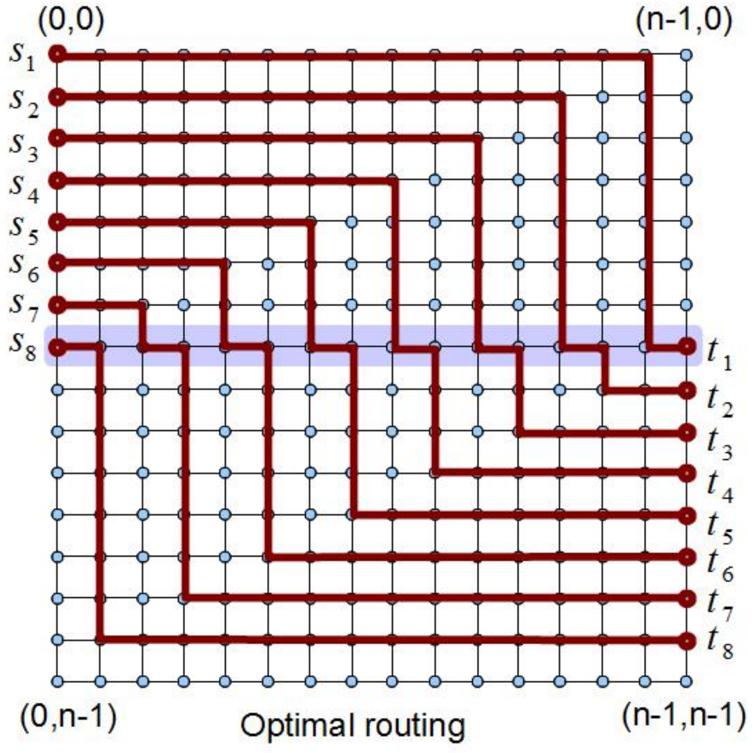
Theorem 10.0.1 *Any basic bottleneck game instance* $\mathbf{R}$ *has at least one Nash Equilibrium and* $PoS(\mathbf{R}) = 1$.

10.1 Price of Anarchy Analysis for Basic Bottleneck Routing Game

Next, we show that there are instances of the basic bottleneck game with large price of anarchy even when β is small.

Theorem 10.1.1 *There is a basic bottleneck game instance* $\mathbf{R}$ *in the* 2*-dimensional grid, with* $\beta = 0(1)$ *bends, such that* $PoA(\mathbf{R}) = \Omega(\sqrt{N})$.

Proof: Consider an $n \times n$ grid. In the game there are $\kappa = n/2$ players, where each player π_i has source s_i in node $(0, i-1)$ of the column 0, and destination t_i in node $(n-1, i+n/2-2)$ of column $n-1$ (see Figure 10.1). The strategy set of player π_i consists of two paths $\mathcal{P}_i = \{p_i^1, p_i^2\}$. Both of the paths cross row $r = n/2 - 1$ (the row is highlighted in Figure 10.1). Path p_i^1 uses one "dedicated" edge in row r, so that the dedicated edges of different players do have any common nodes (see left of Figure 10.1). The remaining path segments of p_i^1 are used to connect the source and destination so that the first strategy paths of the players are disjoint. Note that path p_i^1 consists of at most five path segments (6 bends). Path p_i^2 uses all the edges of row r, and it consists of at most three path segments (4 bends), one in column 0, one in row r, and one in column $n-1$ (see right of Figure 10.1).



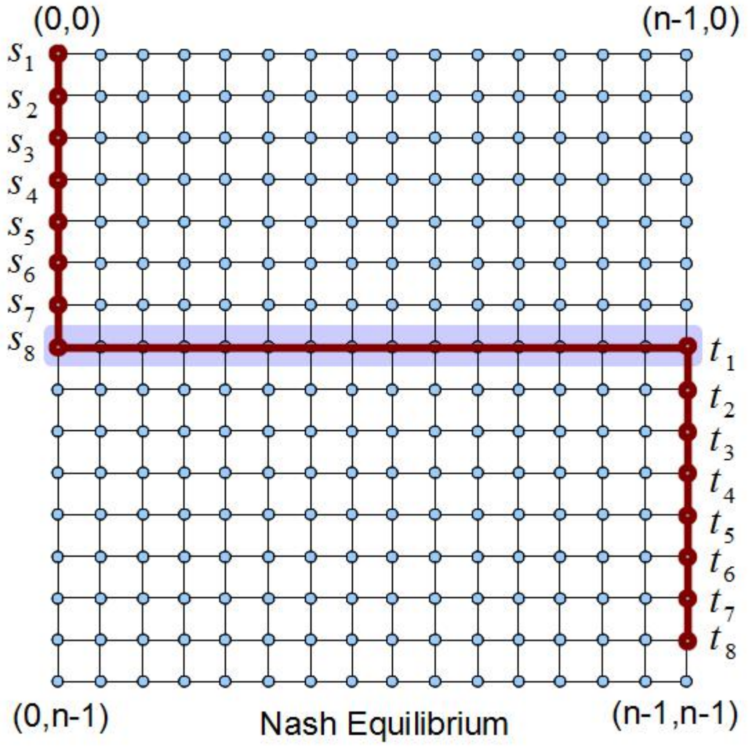


FIGURE 10.1. A game with large price of anarchy and small number of bends

The routing with the first path choices $\mathbf{p}^1 = [p_1^1, p_2^1, \ldots, p_\kappa^1]$ is optimal, since the congestion is $C(\mathbf{p}^1) = 1$. The routing with the second path choices $\mathbf{p}^2 = [p_1^2, p_2^2, \ldots, p_\kappa^2]$ has congestion $C(\mathbf{p}^2) = \kappa$ and every player has cost $pc_i(\mathbf{p}^2) = \kappa$, due to the path segments in row r. Routing $\mathbf{p}^2$ is a Nash equilibrium, since if any player π_i switches to path p_i^1, then its cost remains κ because it still uses the dedicated edge in row r. Therefore:

$$\begin{aligned} PoA(\mathbf{R}) &\geq \frac{C(\mathbf{p}^2)}{C(\mathbf{p}^1)} \\ &= \kappa \\ &= n/2 \\ &= \Omega(\sqrt{N}). \end{aligned}$$

■

An alternative game for showing Theorem 10.1.1, is also when we consider players which have source and destination adjacent to an edge, and any feasible path is in their strategies. In particular, there are as many players as the number of vertical edges (edges in columns), which is $(n-1)n = n^2 - n$. Each player has source on one endpoint and destination the other end-point of a vertical edge. The optimal solution simply uses the edge as path for each player giving congestion one. However, in the equilibrium, each players uses as path the alternative path that is formed by following the remaining edges of a path in the current column and the edges of an adjacent column. See for example Figure 10.2 and the player with source u and destination v. The players of each adjacent columns are involved in one cycle for length $(n-1)2 + 2 = 2n$ edges. Each path in the equilibrium has at most 4 bends. The price of anarchy is $2n - 2 = \Omega(\sqrt{N})$.

10.2 Conclusions

We presented new bottleneck games on multidimensional grids whose price of anarchy is analyzed in terms of the number of bends that the paths are allowed to follow. We found that the price of anarchy is proportional to the number of bends. We also provided game

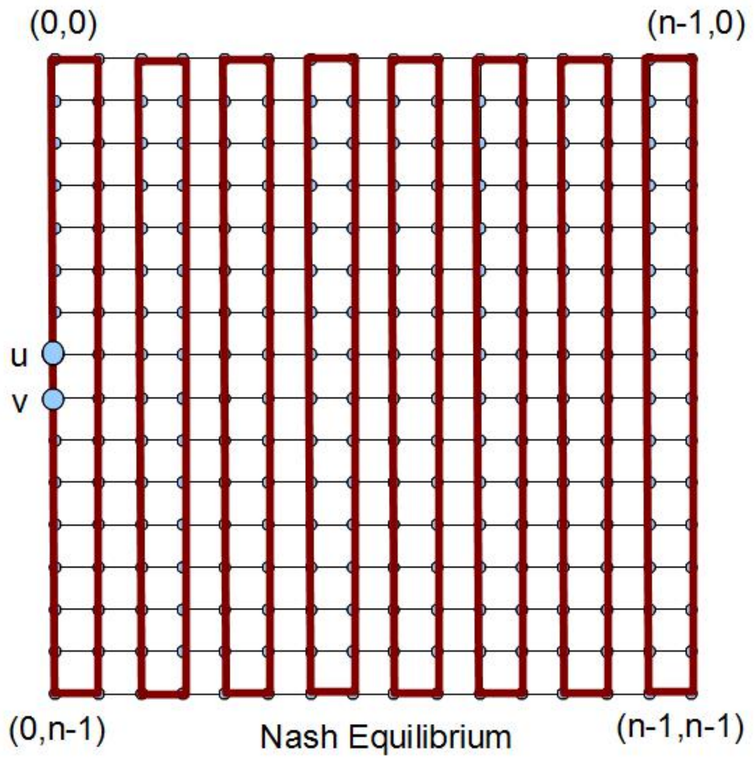


FIGURE 10.2. A alternative game with large price of anarchy and small number of bends

instances that show that the price of anarchy results are tight within poly-log factors. A natural question that remains open is whether we can obtain tighter bounds by removing the poly-log factors. Another interesting problem is to study other network topologies and examine how the notion of bends is generalized in them.

Chapter 11
Channel Routing Games on Grids

11.1 Channel Game

Let $G = (V, E)$ be a d-dimensional grid, with $n^d = N$ nodes. We consider bottleneck routing games where each path is allowed to have at most β bends, and achieve price of anarchy bounded by β. In order to get this price of anarchy we use $\log n$ *channels*, as we describe below.

We can write any path p as a sequence of path segments $p = (q_1, q_2, \ldots, q_k)$, where each q_l is in a line which is in a different dimension than q_{l+1}, where $1 \leq l < k$. The number of nodes (bends) in the path p is bounded by $k + 1 \leq \beta$; thus, the number of path segments is $k \leq \beta - 1$.

Let $\alpha = \log n$. Each edge e accommodates $\alpha = \log n$ distinct *channels* $A_0, A_1, \ldots, A_{\alpha-1}$. The purpose of the channels is to route path segments of different lengths separately. A path segment q whose length is in range $|q| \in [2^j, 2^{j+1} - 1]$ uses channel A_j; we also say that the channel of q is $A(q) = A_j$. Note that a path may use multiple channels according to the lengths of its constituent segments.

Consider a routing $\mathbf{p}$. For any edge e denote by $C_e^{A_j}(\mathbf{p})$ the congestion caused by the path segments of channel A_j, which is equal to the number of path segments in channel A_j that use edge e. The congestion of a path segment q is $C_q(\mathbf{p}) = \max_{e \in q} C_e^{A(q)}(\mathbf{p})$, which is the maximum edge congestion along the path segment and its respective channel. Given a path $p = (q_1, q_2, \ldots, q_k)$, we denote the congestion of the path as the maximum congestion along any of its path segments, namely, $C_p(\mathbf{p}) = \max_{q_i \in p} C_{q_i}(\mathbf{p})$. Using this notion of path congestion all the congestion definitions in Chapter 9 can be extended in a grid with channels.

We are now ready to define the *channel bottleneck game* $\mathbf{R} = (G, \Pi, \mathcal{P})$. As in the basic bottleneck game, there is limit β on the allowed number of bends in a selected path. The social and player cost functions are also similar, $SC(\mathbf{p}) = C(\mathbf{p})$, and $pc_i(\mathbf{p}) = C_{\pi_i}(\mathbf{p}) = C_{p_i}(\mathbf{p})$, where all congestions are calculated using the channel model of the grid. Similar to the basic congestion game we obtain:

Theorem 11.1.1 *Any channel bottleneck game instance* $\mathbf{R}$ *has at least one Nash Equilibrium and* $PoS(\mathbf{R}) = 1$.

11.1.1 Price of Anarchy Analysis for Channel Game

Consider a Nash equilibrium $\mathbf{p} \in \mathcal{P}$. Let $\mathbf{p}^* = [p_1^*, p_2^*, \ldots, p_\kappa^*] \in \mathcal{P}$ be an optimal routing with lowest congestion $C^* = C(\mathbf{p}^*)$. Consider a set of players $\Pi' \subseteq \Pi$ such that the smallest congestion of any player of Π' in routing $\mathbf{p}$ is at least C'. Since $\mathbf{p}$ is an equilibrium, each player $\pi_i \in \Pi'$ has congestion at least $C' - 1$ in its optimal path p_i^*, namely, $C_{p_i^*}(\mathbf{p}) \geq C' - 1$. The $C' - 1$ congestion in p_i^* is due to some path segment $q_i^* \in p_i^*$ with congestion at least $C' - 1$, namely, $C_{p_i^*}(\mathbf{p}) \geq C_{q_i^*}(\mathbf{p}) \geq C' - 1$. Thus, there is an edge $e \in q_i^*$ such that $C_e^{A(q_i^*)} \geq C' - 1$. We call e the *special edge* of player π_i and the respective channel $A(q_i^*)$ the *special channel* of player π_i. Note that a player could have multiple special edges and respective special channels, in which case we choose one of them arbitrarily.

We say that two edges e_1 and e_2 are *far-apart with respect to channel* A_j if the edges are in different dimensions, or if the edges are in the same dimension and in different lines, or if the edges are in the same line and the shortest path length that connects any of their adjacent nodes is at least $2^{j-1} - 1$. If two edges are not far-apart with respect to channel A_j, then we say that they are *close* with respect to channel A_j.

Let $\mathbf{A}(\Pi')$ denote the channel which is special for the majority of the players in Π'. Let $B(\Pi')$ be the subset of players in Π' with special channel $\mathbf{A}(\Pi')$. Clearly, since there are α channels, $|B(\Pi')| \geq |\Pi'|/\alpha$. Let $\Gamma(\Pi')$ denote the set of special edges for the players in $B(\Pi')$. Let $\Delta(\Pi')$ denote a maximum set of edges such that $\Delta(\Pi') \subseteq \Gamma(\Pi')$, and each pair of edges in $\Delta(\Pi')$ is far-apart with respect to channel $\mathbf{A}(\Pi')$. Let $\Phi(\Pi')$ denote the set of players which in routing $\mathbf{p}$ use an edge in $\Delta(\Pi')$ such that the path segment that crosses the edge belongs to channel $\mathbf{A}(\Pi')$. Each player $\pi_i \in B(\Pi')$ has either (i) its special edge $e \in \Delta(\Pi')$, or (ii) there is an edge $e' \in \Delta(\Pi')$ such that e' is close to e with respect to channel $\mathbf{A}$. In either case, we say that player π_i is *assigned* to respective edge e or e' of $\Delta(\Pi')$.

Lemma 11.1.2 *For any set of players $\Pi' \subseteq \Pi$, each edge in $\Delta(\Pi')$ has assigned to it at most $5C^*$ players of Π' in routing $\mathbf{p}$.*

Proof: Suppose that the channel $\Delta(\Pi')$ is in dimension x. Assume that there is an edge $e \in \Delta(\Pi')$ such that there are at least $z \geq 5C^* + 1$ players assigned to it. Let X be the set of players in $B(\Pi')$ which are assigned to e because e is their special edge (case (i) above). Let Y be the number of players in $B(\Pi')$ which are assigned to e because e is near their special edge (case (ii) above). We have that $z = |X| + |Y|$. If $|X| > C^*$, then the edge e is used in the optimal path of at least $C^* + 1$ players, which is impossible since the optimal congestion is C^*. Therefore, $|X| \leq C^*$, and hence $|Y| \geq 4C^* + 1$.

For ease of presentation, assume without loss of generality that x is the horizontal dimension. For any player $\pi_i \in Y$ we say that its special edge is in the *first (second)* part of its optimal path segment if it is positioned in the left (right) half of its optimal path segment (if the special edge is positioned exactly in the middle of the path segment then it is simultaneously in the first and second parts). Let Y_l and Y_r denote the players whose special edges appear on the left and right of e, respectively. Without loss of generality, assume that $|Y_l| \geq |Y|/2$. Without loss of generality, assume also that at least half of the special edges in Y_l are in the first half of their respective optimal segments. Denote by Y_l' these players. We have that $|Y_l'| \geq |Y|/4$. By the positions of the special edges of Y_l' all their optimal path segments intersect, which implies that there is an edge on same line with e which in the optimal routing $\mathbf{p}^*$ has congestion at least

$$|Y_l'| \geq \frac{|Y|}{4} \geq \frac{4C^* + 1}{4} > C^*.$$

This is a contradiction. ■

From Lemma 11.1.2, each edge in $\Delta(\Pi')$ is assigned at most $5C^*$ players of $B(\Pi')$. Since

$$|B(\Pi')| \geq \frac{|\Pi'|}{\alpha},$$

we have:

Corollary 11.1.3 *For any set of players $\Pi' \subseteq \Pi$,*

$$|\Delta(\Pi')| \geq \frac{|\Pi'|}{5\alpha C^*}.$$

Lemma 11.1.4 *For any set of players $\Pi' \subseteq \Pi$ with congestion at least C',*

$$|\Phi(\Pi')| \geq \frac{(C' - 1)|\Pi'|}{20\alpha\beta C^*}.$$

Proof: Each edge in $\Delta(\Pi')$ is special for some player in $B(\Pi')$. Without loss of generality, let $\mathbf{A}(\Pi') = A_j$. Then, $2^{j+1} - 1$ is the maximum path segment of any path that uses channel $\mathbf{A}(\Pi')$. By the definition of the special edges $\Delta(\Pi')$, each path segment of channel $\mathbf{A}(\Pi')$ can have at most four special edges. Since each player in $\Phi(\Pi')$ has at most β path segments each using at most four special edges in $\Delta(\Pi')$, and each special edge in $\Delta(\Pi')$ is used by at least $C' - 1$ players in $\Phi(\Pi')$ (since the edge e has congestion $C' - 1$ in channel $\mathbf{A}(\Pi')$), from Corollary 11.1.3 we obtain:

$$\begin{aligned} |\Phi(\Pi')| &\geq \frac{(C' - 1)|\Delta(\Pi')|}{4\beta} \\ &\geq \frac{(C' - 1)|\Pi'|}{20\alpha\beta C^*}. \end{aligned}$$

■

Theorem 11.1.5 *$C(\mathbf{p}) \leq 40\alpha\beta C^* + \log(5\alpha d n^d C^*)$.*

Proof: Suppose that $C(\mathbf{p}) > 40\alpha\beta C^* + \log(5\alpha d n^d C^*)$. There is a player $\pi_i \in \Pi$ with congestion $C_{\pi_i}(\mathbf{p}) = C(\mathbf{p})$. We define recursively a sequence of player sets $\Pi_0, \Pi_1, \ldots, \Pi_k$, where $k = \log(5\alpha d n^d C^*)$ as follows. We define $\Pi_0 = \{\pi_i\}$. Suppose we have defined the set Π_t, where $t \geq 1$; we define $\Pi_{t+1} = \Phi(\Pi_t)$. From the above definition of Π_t, we have that for each $\pi_j \in \Pi_t$,

$$\begin{aligned} C_{\pi_j}(\mathbf{p}) &\geq C(\mathbf{p}) - t \\ &\geq C(\mathbf{p}) - k \\ &\geq 40\alpha\beta C^* + 1. \end{aligned}$$

From Lemma 11.1.4, $|\Pi_{t+1}| \geq 2|\Pi_t|$. Therefore,

$$|\Pi_k| \geq 2^k \geq 5\alpha d n^d C^*.$$

Consequently, from Corollary 11.1.3,

$$|\Delta(\Pi_k)| \geq \frac{|\Pi_k|}{5\alpha C^*} \geq d n^d.$$

However, we have a contradiction, since $|\Delta(\Pi_k)| \leq |E| < d n^d$. ■

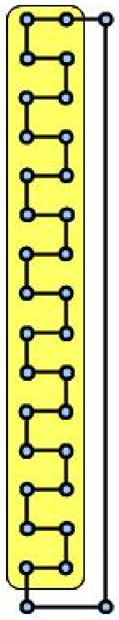
From Theorem 11.1.5, since $\alpha = O(\log n)$ and $N = n^d$, we obtain the following corollary:

Corollary 11.1.6 *For any channel bottleneck game $\mathbf{R}$ in the d-dimensional grid which allows paths with at most β bends, $PoA(\mathbf{R}) = O((\beta/d)\log N)$.*

11.2 Lower Bound

Here, we give lower bounds in terms of bends for the price of anarchy for the channel games.

Theorem 11.2.1 *In the d-dimensional grid with N nodes, given any $\beta \leq c'N$, for a specific constant c', there is a channel bottleneck game instance R with at most β bends, such that $PoA(R) = \Omega(\beta)$.*



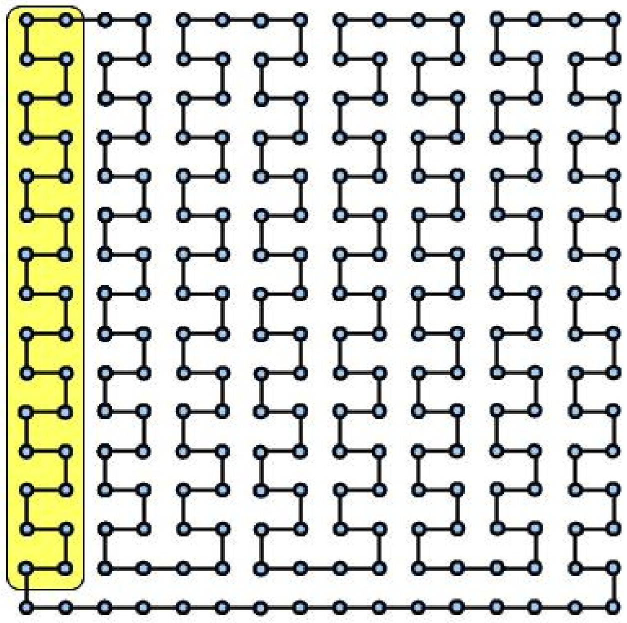


FIGURE 11.1. Zig-zag path and cycles

Proof: We present the result for the 2-dimensional $n \times n$ grid G, and it can be extended to the d-dimensional grid. We define a game along a cycle c of the grid. The main building block of the cycle is the *zig-zag* path which is formed in two consecutive columns, by alternating edges between the columns and rows, as shown highlighted in the left of Figure 12.1. A x-zig-zag path contains x horizontal edges and $x - 1$ vertical edges, giving $2x - 2$ bends (without counting the end nodes). Given an x-zig-zag path we can build a cycle by closing the end points with 4 additional bends, giving a cycle with total $2x + 2$ bends. Since $x < n - 1$ (last row is reserved to close the cycle), the maximum number of bends that a single zig-zag path can provide is bounded by $2(n - 1) + 2 = 2n$.

In order to obtain a cycle with larger number of bends, we combine multiple zig-zag paths, as shown in the middle of Figure 12.1. The largest cycle is formed by using $n/2$ instances of $(n - 1)$-zig-zag paths by combining their original version and their horizontal mirrors, and connecting them with bridge edges in rows 0 and $n - 2$ and closing the loop with a path in row $n - 1$ and bridge edges in the bottoms of columns 0 and $n - 1$. This construction gives a cycle with total

$$\ell = (2(n - 1) + 2) \cdot n/2 + 4 = n^2 + n + 4$$

bends. Using the above construction and adjusting appropriately the sizes of the zig-zag paths it is possible to obtain a cycle with any number of bends β up to ℓ. Clearly, the total number of edges in the cycle is $|c| = \Theta(\beta)$.

We define now a channel bottleneck game $\mathbf{R} = (G, \Pi, \mathcal{P})$. Let Z denote the set of edges in the zig-zag paths, excluding the edges adjacent to the end nodes of each zig-zag path. The game has $\kappa = |Z|$ players $\Pi = \{\pi_1, \dots, \pi_\kappa\}$. Player π_i has two strategy sets: $\mathcal{P}_i = \{p_i^1, p_i^2\}$, where p_i^1 consists only of edge $e_i = (u_i, v_i) \in Z$ in a zig-zag path, and path p_i^2 consists of the alternate path in the cycle from v_i to u_i that traverses all the edges of c except e_i. The edges $e_i \in Z$ are chosen so that different players use different edges. Note that the first path has 2 bends, while the second path has β bends.

The optimal routing $\mathbf{p}^* \in \mathcal{P}$ is the one where each player π_i uses strategy p_i^1, namely, $\mathbf{p}^* = [p_1^1, p_2^1, \ldots, p_\kappa^1]$. The congestion of $\mathbf{p}^*$ is $C(\mathbf{p}^*) = 1$, since edge is used by at most one player. Consider now routing $\mathbf{p} = [p_1^2, p_2^2, \ldots, p_\kappa^2]$, consisting of the second strategy of each player. Routing $\mathbf{p}$ has congestion $C(\mathbf{p}) = \kappa - 1$, since all players except π_i use edge $e_i \in Z$ and all the path segments that use e_i belong to the same channel A_0 for unit length segments. The routing $\mathbf{p}$ is a Nash equilibrium, since if any user π_i attempts to switch to alternate strategy p_i^1, the congestion of the becomes $\kappa + 1 > C(\mathbf{p})$. Therefore we have that:

$$PoA \geq C(\mathbf{p})/C(\mathbf{p}^*) = \kappa - 1 = |Z| - 1 = \Omega(\beta).$$

■

Using similar zig-zag paths for the split model by adjusting appropriately the bend distances (see Figure 12.1) we can obtain the following lower bound:

Theorem 11.2.2 *In the d-dimensional grid with N nodes, given any $\beta \leq c''N$, for a specific constant c'', there is a split bottleneck game instance R with at most β bends, such that $PoA(R) = \Omega(\beta)$.*

11.3 Conclusions

We presented new bottleneck games on multidimensional grids whose price of anarchy is analyzed in terms of the number of bends that the paths are allowed to follow. We found that the price of anarchy is proportional to the number of bends. We also provided game instances that show that the price of anarchy results are tight within poly-log factors. A natural question that remains open is whether we can obtain tighter bounds by removing the poly-log factors. Another interesting problem is to study other network topologies and examine how the notion of bends is generalized in them.

Chapter 12
Split Routing Games on Grids

12.1 Split Game

We describe a way to split the path segments of a path in different lines according to their lengths. In this way we only need to use a single channel that all players can share. For ease of presentation, we first describe the respective game in the 2-dimensional grid, and then explain below how it can be extended to higher dimensions.

Let $G = (V, E)$ be a 2-dimensional $n \times n$ grid. Let $\alpha = \log n$. For convenience take n to be a multiple of $2 \log n$. The odd index rows (columns) $1, 3, \ldots, n-1$ are used to route horizontal (vertical) path segments of lengths ranging from 2 to $n-1$. In particular, row $(2i+1) \bmod \alpha$ (column $2i \bmod \alpha$), where $i \in [0, n/2-1]$, is used for horizontal (vertical) path segments whose length is in range $[2^{i \bmod \alpha}, 2^{(i \bmod \alpha)+1} - 1]$. The even rows (columns) $0, 2, \ldots, n-2$ are reserved to route horizontal (vertical) path segments whose length is in range $[1, 2\alpha - 1]$. Note that path segments in range $[2, 2\alpha - 1]$ have a chance to be routed either in even or odd rows and columns. We say that an odd row (column) $2i+1$ ($2i$) is of *type-($i \bmod \log n$)*, while any even row (column) is of the *local*-type. Note that there are $\alpha + 1$ types in total. Any edge $e \in E$ has the same type of the row or column that it belongs to. Note that with splitting the path segments into different rows we achieved to have a single channel that all players can share.

We are now ready to define the *split bottleneck game* $\mathbf{R} = (G, \Pi, \mathcal{P})$. As in the basic bottleneck game, there is limit β on the number of bends of a path. Each path has to follow the rules for using the appropriate rows and columns for its segments as described above. The social and player cost functions are similar, $SC(\mathbf{p}) = C(\mathbf{p})$, and $pc_i(\mathbf{p}) = C_{\pi_i}(\mathbf{p}) = C_{p_i}(\mathbf{p})$. Similar to the basic congestion game we obtain:

Theorem 12.1.1 *Any split bottleneck game instance* $\mathbf{R}$ *has at least one Nash Equilibrium and* $PoS(\mathbf{R}) = 1$*.*

12.1.1 Price of Anarchy Analysis for Split Game

Consider a Nash equilibrium $\mathbf{p} \in \mathcal{P}$. Consider a set of players $\Pi' \subseteq \Pi$. We can define the special edge and special type for a player in the same way as we did for channel bottleneck games. The only difference is that instead of the notion of the channel we use the notion of the type. Let $\tau(\Pi')$ be the type which is special for the majority of the players in Π'. Using $\tau(\Pi')$ we can define the sets: $B(\Pi')$, $\Gamma(\Pi')$, $\Delta(\Pi')$, and $\Phi(\Pi')$, as we did in Section 11.1, where $\tau(\Pi')$ plays the role of $\mathbf{A}(\Pi')$. We have that

$$|B(\Pi')| \geq |\Pi'|/(\alpha + 1),$$

since there are $\alpha + 1$ types.

Lemma 12.1.2 *For any set of players* $\Pi' \subseteq \Pi$*, each edge* $e \in \Delta(\Pi')$ *has assigned to it at most* $c_1 \alpha C^*$ *players of* Π' *in routing* $\mathbf{p}$*, for some constant* c_1*.*

Proof: Let $Z \subseteq \Pi'$ denote the set of players that are assigned to e. We examine the following cases:

- e is of the local type:

  Let $e = (u, v)$, and without loss of generality assume the e is horizontal. Each of the players in Z must have e as a special edge or their special edge is close to e. Let Y denote the set of local type edges which are close to e. The edges in Y are in the same row as e and one of their end nodes is at distance at most $2\alpha - 2$ from u or v. For convenience we include e in A. The number of edges in A (including e) is at most $4\alpha - 1$. Each special edge can accommodate at most C^* optimal paths from the players in Z. Therefore, $|Z| \leq C^* \cdot (4\alpha - 1)$.

- e is of type-k:

  The analysis if the same as in Lemma 11.1.2, where we obtain that $|Z| \leq 5C^*$.

■

From Lemma 12.1.2, each edge in $\Delta(\Pi')$ is assigned at most $c_1\alpha C^*$ players of $B(\Pi)$. Since

$$|B(\Pi')| \geq \frac{|\Pi'|}{\alpha+1},$$

we have

$$|\Delta(\Pi')| \geq \frac{|\Pi'|}{(\alpha+1)\cdot(c_1\alpha C^*)}.$$

Therefore,

Corollary 12.1.3 *For any set of players* $\Pi' \subseteq \Pi$*,*

$$|\Delta(\Pi')| \geq \frac{|\Pi'|}{c_2\alpha^2 C^*},$$

for some constant c_2*.*

Lemma 12.1.4 *For any set of players* $\Pi' \subseteq \Pi$ *with congestion at least* C'*,*

$$|\Phi(\Pi')| \geq \frac{(C'-1)|\Pi'|}{c_3\alpha^2\beta C^*},$$

for some constant c_3*.*

Proof: Each edge in $\Delta(\Pi')$ is special for some player in $B(\Pi')$. By the definition of the special edges $\Delta(\Pi')$, if $\tau(\Pi')$ is equal to type-i, then each path segment can use at most four special edges in $\Delta(\Pi')$. Otherwise, if $\tau(\Pi')$ is the local type, then each path segment can use at most one special edge. Since each player in $\Phi(\Pi')$ has at most β path segments each using at most four special edges in $\Delta(\Pi')$, and each special edge in $\Delta(\Pi')$ is used by at least $C'-1$ players in $\Phi(\Pi)$, from Corollary 12.1.3 we obtain:

$$\begin{aligned} |\Phi(\Pi')| &\geq \frac{(C'-1)|\Delta(\Pi')|}{4\beta} \\ &\geq \frac{(C'-1)|\Pi'|}{4c_2\cdot\alpha^2\beta C^*}. \end{aligned}$$

■

Theorem 12.1.5 $C(\mathbf{p}) \leq 2c_3\alpha^2\beta C^* + \log(2c_2\alpha^2 n^2 C^*)$.

Proof: Suppose that $C(\mathbf{p}) > 2c_3\alpha^2\beta C^* + \log(2c_2\alpha^2 n^2 C^*)$. There is a player $\pi_i \in \Pi$ with congestion $C_{\pi_i}(\mathbf{p}) = C(\mathbf{p})$. We define recursively a sequence of player sets $\Pi_0, \Pi_1, \ldots, \Pi_k$, where $k = \log(2c_2\alpha^2 n^2 C^*)$ as follows. We define $\Pi_0 = \{\pi_i\}$. Suppose we have defined the set Π_t; we define $\Pi_{t+1} = \Phi(\Pi_t)$. From the above definition of Π_t, we have that for each $\pi_j \in \Pi_t$,

$$\begin{aligned} C_{\pi_j}(\mathbf{p}) &\geq C(\mathbf{p}) - t \\ &\geq C(\mathbf{p}) - k \\ &\geq 2c_3\alpha^2\beta C^* + 1. \end{aligned}$$

From Lemma 12.1.4, $|\Pi_{t+1}| \geq 2|\Pi_t|$. Therefore,

$$|\Pi_k| \geq 2^k = 2c_2\alpha^2 n^2 C^*.$$

Consequently, from Corollary 12.1.3,

$$\begin{aligned} |\Delta(\Pi_k)| &\geq \frac{|\Pi_k|}{c_2\alpha^2 C^*} \\ &\geq 2n^2. \end{aligned}$$

However, we have a contradiction, since $|\Delta(\Pi_k)| \leq |E| < 2n^2$. ■

From Theorem 12.1.5, since $\alpha = \log n$ and $N = n^2$, we obtain the following corollary:

Corollary 12.1.6 *For any split bottleneck game* $\mathbf{R}$ *in the* 2*-dimensional grid which allows paths with at most* β *bends,* $PoA(\mathbf{R}) = O(\beta \log^2 N)$.

12.1.2 Split Game in the d-Dimensional Grid

We can extend the split games to a grid with d dimensions. The first dimension takes the role of the horizontal dimension, and the second dimension takes the role of the vertical dimension. Any other dimension (third and above) uses the first dimension to split the path segments. For example, in the 3-dimensional grid, a path segment q in the third dimension is a sequence of nodes with coordinates $q = (x, y, z), \ldots, (x, y, z + k)$. This path segment is placed in an appropriate odd first coordinate $x = 2^i + 1$ if $k \in [2^{i \bmod \alpha}, 2^{(i \bmod \alpha)+1} - 1]$, and if $k \leq 2\alpha - 1$ then it could use an even first coordinate $x = 2^i$. In this way we can characterize q as *type-i*, or local type, respectively. The total number of types for the d-dimensional grid remains $\alpha + 1$.

The main difference in the price of anarchy analysis is that Theorem 12.1.5 now returns $C(\mathbf{p}) \leq 2c_3\alpha^2\beta C^* + \log(c_2\alpha^2 d n^d C^*)$. Since $\alpha = O(\log n)$ and $N = n^d$, Corollary 12.1.6 now becomes:

Corollary 12.1.7 *For any split bottleneck game* $\mathbf{R}$ *in the d-dimensional grid which allows paths with at most* β *bends,* $PoA(\mathbf{R}) = O((\beta/d^2)\log^2 N)$.

12.2 Lower Bound

Here, we give lower bound in terms of bends for the price of anarchy for the split games.

Using similar zig-zag paths for the split model by adjusting appropriately the bend distances (see right of Figure 12.1) we can obtain the following lower bound:

Theorem 12.2.1 *In the d-dimensional grid with* N *nodes, given any* $\beta \leq c''N$, *for a specific constant* c'', *there is a split bottleneck game instance* R *with at most* β *bends, such that* $PoA(R) = \Omega(\beta)$.

Proof: The proof is similar to Theorem 11.2.1. The main difference is that we use zig-zag paths on even rows and columns which allow the use of short length path segments of the local type (see right part of Figure 12.1). Thus, each zig-zag edge is now a path segment of length 2. All the bridge edges are also in the even rows and columns. The lowest path segment that connects the leftmost and rightmost zig-paths needs to use an odd row which allows the particular long path length. The odd row can be reached with a vertical path segment of length at most 3 from the leftmost and rightmost zig-zag paths. Therefore, an x-zig-zag path contains x horizontal path segments, which is bounded as $x \leq (n-3)/2 = n/2 - 3/2$, and

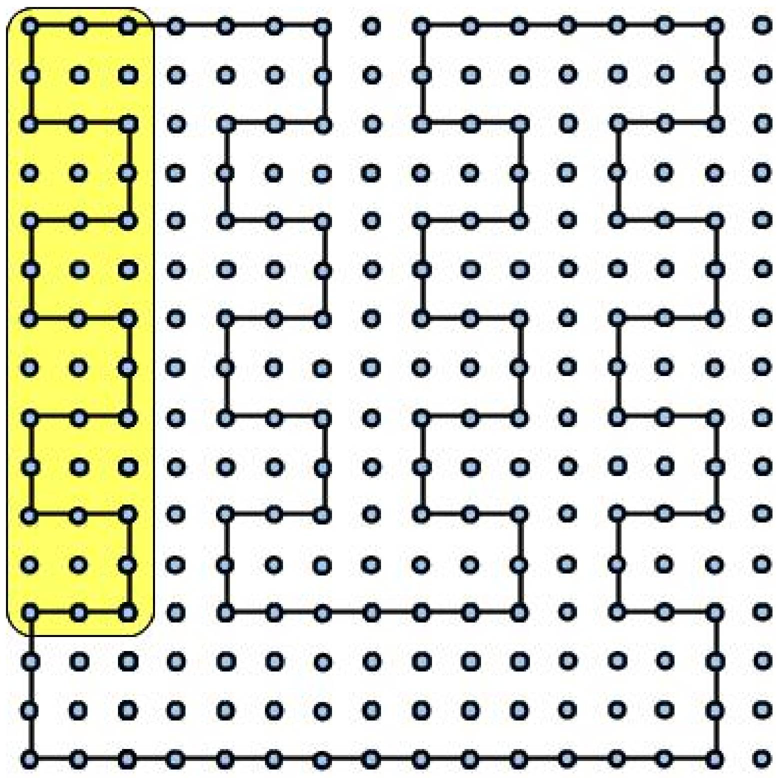


FIGURE 12.1. Zig-zag path and cycles

consists of $2x + 2 \leq n - 1$ bends. This implies that the largest cycle has number of bends:

$$\ell = (n-1) \cdot n/4 + 4 = n^2/4 - n/4 + 4.$$

Now each player's p_i^1 strategy is a path segment of length 2 in a zig-zag path (which is of the local type), while path p_i^2 is the alternate path in the cycle. The asymptotic analysis on the price of anarchy remains the same as in Theorem 11.2.1. ■

12.3 Conclusions

We presented new bottleneck games on multidimensional grids whose price of anarchy is analyzed in terms of the number of bends that the paths are allowed to follow. We found that the price of anarchy is proportional to the number of bends. We also provided game instances that show that the price of anarchy results are tight within poly-log factors. A natural question that remains open is whether we can obtain tighter bounds by removing the poly-log factors. Another interesting problem is to study other network topologies and examine how the notion of bends is generalized in them.

Chapter 13
Conclusions to Bottleneck Routing Games on Grids

In this dissertation we also considered routing games on grid network topologies. We showed that the price of anarchy in bottleneck games in grids is proportional to the number of bends β that the paths are allowed to take in the grids' space. We presented games where the price of anarchy was $\widetilde{O}(\beta)$. We also give respective lower bound of $\Omega(\beta)$ which shows that our upper bound is within only a poly-log factor from the best achievable price of anarchy. A significant impact of our analysis is that there exist bottleneck routing games with small number of bends which give a poly-log approximation to the optimal coordinated solution that may use an arbitrary number of bends. To our knowledge, this is the first tight analysis of bottleneck games on grids.

Further work in this area could involve the analysis of routing games on other more complex topologies like mesh networks.

References

[1] DJ Abraham and A Blum. Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges. *of the 8th ACM conference on*, 2007.

[2] E. Anshelevich, S. Das, and Y. Naamad. Anarchy, stability, and utopia: Creating better matchings. *Algorithmic Game Theory*, pages 159–170, 2009.

[3] Esther M. Arkin and Raphael Hassin. On Local Search for Weighted k-Set Packing. *Mathematics of Operations Research*, 23(3):640–648, August 1998.

[4] R.J. Aumann. Values of Non-Atomic Games, Part V: Monetary Economies. Technical report, 1971.

[5] Vineet Bafna, Babu Narayanan, and R Ravi. Nonoverlapping Local Alignments (Weighted Independent Sets of Axis Parallel. *Science*, (May 1996):1–15.

[6] MF Balcan, S Krehbiel, and G Piliouras. Near optimality in covering and packing games by exposing global information. *Arxiv preprint arXiv:*, pages 1–16, 2011.

[7] Ron Banner and Ariel Orda. Bottleneck routing games in communication networks. *IEEE Journal on Selected Areas in Communications*, 25(6):1173–1179, 2007. also appears in INFOCOM'06.

[8] Costas Busch, Rajgopal Kannan, and Athanasios V. Vasilakos. Quality of routing congestion games in wireless sensor networks. In *Proc. 4th International Wireless Internet Conference (WICON)*, Maui, Hawaii, November 2008.

[9] Costas Busch and Malik Magdon-Ismail. Atomic routing games on maximum congestion. *Theoretical Computer Science*, 410(36):3337–3347, August 2009.

[10] Costas Busch, Malik Magdon-Ismail, and Jing Xi. Optimal oblivious path selection on the mesh. *IEEE Trans. Computers*, 57(5):660–671, 2008.

[11] Barun Chandra and Magnús M Halldórsson. Greedy Local Improvement and Weighted Set Packing Approximation. *Journal of Algorithms*, 39(2):223–240, May 2001.

[12] George Christodoulou and Elias Koutsoupias. The price of anarchy of finite congestion games. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 67–73, Baltimore, MD, USA, May 2005. ACM.

[13] J. Edmonds. Paths, trees, and flowers. *Classic Papers in Combinatorics*, pages 361–379, 1987.

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

[15] Tobias Harks, Max Klimm, and Rolf H. Möhring. Strong nash equilibria in games with the lexicographical improvement property. In *WINE '09: Proceedings of the 5th International Workshop on Internet and Network Economics*, pages 463–470, Berlin, Heidelberg, 2009. Springer-Verlag.

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

[17] Viggo Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, pages 1–12, 1991.

[18] Rajgopal Kannan and Costas Busch. Bottleneck congestion games with logarithmic price of anarchy. In *Proc. 3rd Annual Symposium on Algorithmic Game Theory (SAGT 2010)*, October 2010. To appear.

[19] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1563 of *LNCS*, pages 404–413, Trier, Germany, March 1999. Springer-Verlag.

[20] David Leibovic. Selfish Set Covering. *Midstates Conference for Undergraduate*, 2009.

[21] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-scheduling in $O(congestion + dilation)$ steps. *Combinatorica*, 14:167–186, 1994.

[22] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes.* Morgan Kaufmann, San Mateo, 1992.

[23] Lavy Libman and Ariel Orda. Atomic resource sharing in noncooperative networks. *Telecomunication Systems*, 17(4):385–409, 2001.

[24] D. Monderer and L. S. Shapely. Potential games. *Games and Economic Behavior*, 14:124–143, 1996.

[25] Christos Papadimitriou. Algorithms, games, and the Internet. In ACM, editor, *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 749–753, Hersonissos, Crete, Greece, July 2001.

[26] R. W. Rosenthal. A class of games possesing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[27] Tim Roughgarden. The maximum latency of selfish routing. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 980–981, New Orleans, Louisiana, (USA), January 2004.

[28] Tim Roughgarden. Selfish routing with atomic players. In *Proc. 16th Symp. on Discrete Algorithms (SODA)*, pages 1184–1185. ACM/SIAM, 2005.

[29] Tim Roughgarden and Éva Tardos. How bad is selfish routing. *Journal of the ACM*, 49(2):236–259, March 2002.

[30] Tim Roughgarden and Éva Tardos. Bounding the inefficiency of equilibria in nonatomic congestion games. *Games and Economic Behavior*, 47(2):389–403, 2004.

[31] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case. *Artificial Intelligence*, 111:209–238, 1999.

[32] Tuomas W Sandholm and Victor R Lesser. Coalitions among Computationally Bounded Agents. *Science*, pages 1–44.

[33] L. Shapley and H. Scarf. On cores and indivisibility. 1973.

[34] Subhash Suri, Csaba D. Toth, and Yunhong Zhou. Selfish load balancing and atomic congestion games. *Algorithmica*, 47(1):79–96, January 2007.

[35] JR Votano, M Parham, and LH Hall. Coalition formation among autonomous agents: Strategies and complexity. *Chemistry*, 2004.

Vita

Alfred B. Samman was born on March, in picturesque Accra, Ghana. He finished his undergraduate studies at the University of Botswana on May 2003. He earned a Master of Science degree in Computer Science from Southern University in May 2006. In August 2007 he came to Louisiana State University to pursue graduate studies in Computer Science. He is currently a candidate for the degree of Doctor of Philosophy in Computer Science, which will be awarded in August 2012.