Order Number 9312205

A generalization of the stable marriage algorithms involving group preferences in resource allocation problems

Li, Zhi Cheng, Ph.D.

University of Hawaii, 1992

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

A GENERALIZATION OF THE STABLE MARRIAGE ALGORITHMS

INVOLVING GROUP PREFERENCES

IN RESOURCE ALLOCATION PROBLEMS

A DISSERTATION SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAI'I IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMMUNICATION AND INFORMATION SCIENCES

DECEMBER 1992

By

Zhi Cheng Li

Dissertation Committee:

Stephen Y. Itoga, Chairperson
Dale W. Myers
W. Wesley Peterson
William E. Remus
Kazuo Sugihara

Acknowledgements

My sincere thanks to all who made this dissertation possible.

- To my adviser, Stephen Y. Itoga, for introducing me to the stable marriage problems, for allowing me the freedom to choose research topics, for helping me to identify the research directions, for the support I needed to finish this dissertation.

- To Professors William Wesley Peterson, William E. Remus and Kazuo Sugihara, for reviewing this thesis and providing advice and suggestions, for the lessons, support and encouragement during my PhD study, without which this thesis would be impossible.

- To Professor Dale W. Myers, for many hours and days he spent to discuss this thesis, to check the proofs, and to provide advice and suggestions, for insisting on high standards on this disseration, for being a good teacher and a good friend. I am especially grateful to his intellectual challenges and support.

- To my mother, my father, my wife, my son and my sisters, for their constant support and care.

- This research was partly supported by the National Science Foundation under the grant IRI 89-13678.

Abstract

The resource allocation problem usually involves optimization on uniform resources. This thesis solves the problem of finding matchings between several resources and activities which are optimal for a given linear objective function and are stable with regard to preferences of the resources and activities. The concept of a stable matching comes from the stable marriage problem. Given an equal number of men and women, and, for each person, a strictly ordered preference list containing all the members of the opposite sex, a stable marriage is a one-to-one matching of men and women in which there is no man and woman who prefer each other to their partners. Although this stable marriage problem with its strictly ordered preference lists (no ties or indifference) has been studied for three decades in computer science, there are few applications. When arbitrary indifference is allowed, the stable marriage problem has more applications but fails to have some valuable properties such as Pareto Efficiency and majority assignment. We suggest using group preference lists to allow limited indifference in applications. That is, each person belongs to a group that has a preference list for all groups of the opposite sex. We show that with this approach, stability guarantees Pareto Efficiency. We have constructed a correct definition of majority assignment for our case, and show that stability also guarantees majority assignment. For our approach, there is also a polynomial-size representation of all the stable matchings for a given problem. Thus polynomial-time complexity algorithms are possible for optimization over all such matchings. We give algorithms to find the maximal elements in the solution lattice, to find the polynomial-size representation of the solution lattice using rotations, and to find the partial order on the set of rotations. The first two algorithms generalize those of Gale-Shapley and Gusfield. The third one is new. All three algorithms run in $O(Np)$ time, where N is the total number of men or women, p is the number of groups in the other sex. We also develop the lattice structure theory supporting the algorithms.

Table of Contents

Chapter 1. Introduction

Resource allocation problems are generally defined in terms of resources that should be allocated to activities in such a way as to maximize an objective function or a performance measure selected by the decision maker [18, 10]. Many studies have been done for the optimization problem with a single discrete or continuous resource and a single constraint. In these studies resources are treated as objects without their own preferences for the activities, only the decision maker's benefits are considered. This thesis tries to solve the optimization problem with a linear objective function and with several available discrete resources, in which the mutual preferences of resources and activities have been considered. Hence, one important performance measure in this resource allocation problem is stability.

The concept of stability comes from the stable marriage problem. Given an equal number of men and women, and, for each person, a strictly ordered preference list containing all the members of the opposite sex, a stable marriage is a one-to-one matching of men and women in which there is no pair consisting of a man and a woman such that they prefer each other over their own partners. Since Gale and Shapley introduced this problem and an algorithm for it in 1962 [3], much research has been done on its algorithms [17, 14, 16, 2], on the structure of the space of its solutions [11, 9, 7, 8], and on extensions of the problem [12, 13, 1]. Although this problem and algorithms for this problem have been used in such applications as the hospital and residents problem and college admission problems, the requirement for strictly ordered preference lists for all members of the opposite sex was always a hindrance to its wider application [20].

Gärdenfors [4] suggested allowing groups of persons in one's preference list to indicate one's indifference to ordering of people in the group. His research shows that, unlike the standard case, stable marriages with this variation of the preference lists may lose their advantages over unstable ones because a majority of persons do not necessarily have better partners in such a stable marriage than in an unstable one. Besides, stability in this case does not guarantee the highest possible satisfaction for the matching. There is still room for people to exchange their partners under the stability requirement and obtain more preferred partners. This is a violation to the Pareto Efficiency property: a stable marriage is Pareto Efficient if and only if there are no two married couples such that at least one person prefer swapping partners and the others do not care [4]. Gärdenfors did not suggest a direct way to find all solutions with Pareto Efficiency. Instead, he transforms his more general problem into many standard ones by ordering the indifference groups in all possible ways, finding solutions for each of them, and checking Pareto Efficiency for each solution. This scheme is very inefficient since the number of ways to order the indifference groups is exponential in the size of the groups. However, a greater hindrance to a wider application of his scheme is the lack of structural representation of its solution space without which efficient optimization over the whole solution space is impossible since the number of solutions for a standard stable marriage problem is known to grow exponentially with the problem size.

We suggest using group preference lists to allow limited indifference in applications. That is, each person belongs to a group that has a preference list for all groups of the opposite sex. Our research shows that with this approach stability guarantees Pareto Efficiency. After constructing the correct definition of majority approval for our case, we show that stability guarantees majority approval. For our approach, there is also a polynomial-size representation of the solution space. Thus polynomial-time complexity algorithms are possible for optimization over all stable matchings of a given problem. There are three major algorithms in our approach which find the maximal elements in the solution lattice, find the polynomial-size representation of the solution lattice using rotations, and find the partial order on the set of rotations. The first two algorithms are generalizations of algorithms developed by Gale-Shapley and Gusfield, respectively. The third one is new. All three algorithms run in $O(Np)$ time, where N is the total number of men or women and p is the number of groups in the other sex.

In the next chapter we formally define the problem this thesis addresses. In Chapter 3 we introduce the basic notation and preliminary definitions. Chapter 4 presents the generalized Gale-Shapley Algorithm (Algorithm I) and its correctness proof. In Chapter 5 we discuss Pareto Efficiency. Chapter 6 proves that stability implies majority assignments in our case. Chapters 7 and 8 describe the structure of the solution space for our case. In Chapter 9 we show how to represent all solutions for our case and introduce Algorithm II, which is a generalized version of Gusfield's, for finding all rotations for a grouped stable matching instance. Chapter 10 discusses the partial order on the rotations. In this chapter we introduce Algorithm III, which differs from Gusfield's, for finding the partial order of rotations in our case. Chapter 11 shows how to reduce three optimization problems to our problem model and how to reduce our problem to a minimum cut problem. Chapter 12 contains the conclusions.

Chapter 2. Definition of the Problem

Suppose we try to match two sets, *Green* and *Red*. The sizes of these two sets are not necessarily equal. Each member of the sets is called a slot. We use a small letter, g or r, to denote a slot of the *Green* or the *Red*, respectively. The set *Green* has been partitioned into m non-overlapping subsets, $G_1, G_2, ..., G_m$, and the set *Red* into n non-overlapping subsets, $R_1, R_2, ..., R_n$. All slots within each subset are regarded as identical. We use s_i and t_j to denote the size of subset G_i and R_j respectively. Let $N = \{0, 1, 2, 3, ...\}$.

Now given the following items:

- m; n; s_i for $i \in [1,m]$ such that $s_i \geq 1$, $s_i \in N$; t_j for $j \in [1,n]$ such that $t_j \geq 1$, $t_j \in N$.
- The *preference lists* of each $G_i(R_j)$ over all R_j's(G_i's) in the descending order of preference. The unacceptable partners are dropped from each other's list.
- An *objective matrix*, B, in which each element b_{ij} ($i \in [1,m]$, $j \in [1,n]$), a non-negative real constant, represents the objective value of a matched pair (g, r) where $g \in G_i$, $r \in R_j$.

We want to find a one-to-one, partial matching M between *Green* and *Red*, subject to the following performance criteria. M is represented by a $(m+1)x(n+1)$ matrix A, in which A_{ij} ($i \in [1,m]$, $j \in [1,n]$) is the number of matched slots between the subsets G_i and R_j in M, $A_{m+1,j}$ ($j \in [1,n]$) and $A_{i,n+1}$ ($i \in [1,m]$) the number of unmatched R_j and G_i slots respectively, and $A_{m+1,n+1}$ is defined to be zero. In fact, by the definition of A, we have created two dummy subsets, G_{m+1} and R_{n+1}, with unlimited sizes. G_{m+1} (R_{n+1}) is placed at the end of every R_j's (G_i's) preference list because a matched status is more preferred to a non-matched status by the nature of the optimization problem. $A_{ij} \epsilon N$, for any $i \epsilon [1, m+1]$ and $j \epsilon [1, n+1]$. The performance criteria are:

(1) The size limits are satisfied. That is

$$\sum_{i=1}^{m+1} A_{ij} = t_j \; ; \qquad \sum_{j=1}^{n+1} A_{ij} = s_i \; .$$

(2) M is stable. That means, there is no pair g, r in M such that:

a. g is an element of *Green* and r an element of *Red* that can accept each other;

b. g and r are not matched to each other in M;

c. Each of g and r is either free or strictly prefers the other to its partner in M;

(3) M is Pareto Efficient. That means,

if for any pairs (g_1, r_1) and (g_2, r_2) in M, r_1 ranks g_1 and g_2 the same, and r_2 ranks g_1 and g_2 the same, then not both (1) g_1 ranks r_2 at least as good as r_1, and (2) g_2 prefers r_1 to r_2.

And the other way around, where g and r have replaced each other.

(4) The value of the following objective function is to be maximized:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} (b_{ij} \, A_{ij}) \; .$$

Chapter 3. Basic Notation and Definitions

Green, *Red*: two sets to be matched. Their sizes are not necessarily to be the same. Each member of the sets is called a slot. We use S to denote the set of all slots, and use a small letter, g or r, to denote a slot of the *Green* or the *Red*, respectively. The set *Green* has been partitioned into m disjoint subsets, G_1, G_2, ..., G_m, and the set *Red* into n disjoint subsets, R_1, R_2, ..., R_n.

$G(R)$: a subset of the given partition of *Green*(*Red*). We use subscripts to distinguish them, such as G_i, R_j, etc. All slots within each subset are regarded as identical. Each subset has its own fixed size. We use s_i and t_j to denote the size of subset G_i and R_j respectively.

$g(r)$: a slot of $G(R)$. We use subscripts, such as g_1, r_2, etc, whenever we need to distinguish them. Unless specified otherwise, $g_i(r_j)$ indicates the slot's membership in $G_i(R_j)$, and denotes *any* slot in $G_i(R_j)$.

preference list: the preference list of all slots in $G(R)$ over all *Red* (*Green*) subsets in the descending order of preference. We add a dummy subset G_{m+1} (R_{n+1}) to the end of each R's (G's) list to indicate the unmatched situation (there is a general assumption that a matched status is preferred to a non-matched status). All partners on the preference list are regarded as *acceptable*, partners not on the preference list are regarded as *unacceptable*. We also use $Y \prec_X Z$ to indicate X prefers Y to Z, $Y =_X Z$ to indicate X ranks Y and Z the same, and $Y \preceq_X Z$ to indicate X ranks Y at least as good as Z.

dummy slot: a slot in G_{m+1} or R_{n+1}, created for convenience of representation. In this way, all the original slots are considered to be matched. A free *Green* (*Red*) slot is matched to a dummy slot of R_{n+1} (G_{m+1}). There is no matching between the dummy slots.

A matching instance: a set of slots S and a preference list for each G_i and R_j ($i \in [1,m]$, $j \in [1,n]$).

M: a matching between *Green* slots and *Red* slots. A matching is a one-to-one, partial function from the *Green* slots to the *Red* slots.

equivalent matchings: For a matching instance, two matchings M and M' are equivalent iff they are isomorphic, i.e., there is an isomorphism h which carries from M to M', i.e. h is an one-to-one function from the set of slots to itself such that (1) for any $G \in Green$, for any $g \in G$, $h(g) \in G$; (2) for any $R \in Red$, for any $r \in R$, $h(r) \in R$; and (3) $(g, r) \in M$ iff $(h(g), h(r)) \in M'$. This equivalence is possible since we consider all slots within each subset as identical in terms of their preferences and ranks.

A: a matrix representing a set of equivalent matchings for a matching instance. A is a $(m+1)x(n+1)$, non-negative integer matrix. If M is a matching represented by A, then A_{ij} ($i \in [1,m]$, $j \in [1,n]$) is the number of matched slots between the subsets G_i and R_j in M, $A_{m+1,j}$ ($j \in [1,n]$) and $A_{i,n+1}$ ($i \in [1,m]$) the number of unmatched R_j and G_i slots respectively, and $A_{m+1,n+1}$ is defined to be zero.

Generally, a <u>*matrix for a matching instance*</u> is any non-negative integer matrix satisfying the following condition:

$$\sum_{i=1}^{m+1} A_{ij} = t_j \ ; \ \sum_{j=1}^{n+1} A_{ij} = s_i \ .$$

The operations on matrices, such as, $\min(A, B)$, $A+B$, A-B, follow the standard definitions.

LEMMA 3.1. *Two matchings for an instance are equivalent iff their matrices are the same.*

PROOF. Suppose the matrices A and A' represent the matchings M and M' for a given matching instance, respectively, we want to prove M and M' are equivalent iff $A \equiv A'$.

$=>$: Suppose M and M' are equivalent, then, by definition, there is an isomorphism h on the set of slots which satisfies (1) for any $G \epsilon Green$, for any $g \epsilon G$, $h(g) \epsilon G$; (2) for any $R \epsilon Red$, for any $r \epsilon R$, $h(r) \epsilon R$; and (3) $(g, r) \epsilon M$ iff $(h(g), h(r)) \epsilon M'$. These conditions guarantee that, if there are A_{ij} pairs in M between G_i and R_j, then there are exactly $A'_{ij}=A_{ij}$ pairs in M' between G_i and R_j and vice versa.

$<=$: On the other hand, if $A \equiv A'$, we want to prove there is an isomorphism to satisfy the definition of the equivalent matchings. Suppose p is the maximum number of pairs possible for the instance. We will use induction on $k \epsilon [1,p]$, the total sum of A_{ij}'s (i. e. the total number of pairs). If $k=1$, then for some i and j, $A_{ij}=A'_{ij}=1$ while all other elements of the two matrices are zeros. By the definition of the matrices, there is only one pair each, (g, r) and (g', r'), in M and M', such that $g, g' \epsilon G_i$ and $r, r' \epsilon R_j$. Let $h_1(g)=g'$, $h_1(r)=r'$, $h_1(g')=g$, $h_1(r')=r$ and $h_1(x) = x$ for any other slot x. Then h_1 is an isomorphism and M and M' are equivalent. Suppose the assumption we want to prove is true for $k=t\epsilon[1, p\text{-}1]$. We want to prove it is also true for $k=t+1$. Find a pair (g, r) in M and a pair (g', r') in M' such that $g, g'\epsilon G_i$ and $r, r'\epsilon R_j$. Then we have an isomorphism h_1 on the set of slots as defined above. By the inductive hypothesis, there is another isomorphism h_2 on the set of slots for the remaining t pairs. Obviously the isomorphism $h_1 \circ h_2$ satisfies the definition of the equivalent matchings. Therefore, M and M' are equivalent.

Definition. For any subset X and any matching represented by A, $worst(X, A)$ is the lowest ranked partner of X in the matching represented by A.

Definition. A pair (g, r) is a *blocking pair* for a matching M iff both g and r accept each other, but they are not a pair in M, and each of them strictly prefers the other to its partner in M.

Definition. A pair (g_i, r_j) $(i\epsilon[1, m], j\epsilon[1, n])$ is a *blocking pair* for a matrix for a given matching instance iff both G_i and R_j accept each other, and they prefer each other to their lowest ranked partners, i.e.,

$$\exists i'\epsilon[1,m+1](i'\neq i \wedge A_{i'j}>0 \wedge G_i \prec_{R_j} G_{i'}) \wedge \exists j'\epsilon[1,n+1](j'\neq j \wedge A_{ij'}>0 \wedge R_j \prec_{G_i} R_{j'}).$$

Definition. A matching or a matrix is *stable* iff it has no blocking pair.

Definition. For matrices an alternate definition of stability is

$$\forall i\epsilon[1,m]\forall j\epsilon[1,n]\forall i'\epsilon[1,m+1]\forall j'\epsilon[1,n+1]$$

$$G_{i'} \prec_{R_j} G_i \vee (G_i \prec_{R_j} G_{i'} \wedge A_{i'j}=0) \vee R_{j'} \prec_{G_i} R_j \vee (R_j \prec_{G_i} R_{j'} \wedge A_{ij'}=0).$$

Definition. For any matching instance with a linear ordering $\leq$ of its slots, an *ordered matching* for the instance is a matching such that for any slots a and b of a subset G,

(a) a matched and b unmatched implies $a \leq b$,

(b) G prefers a's partner to b's partner implies $a \leq b$,

(c) if a and b's partners belong to the same R then a's partner $\leq$ b's partner iff $a \leq b$.

Similarly for any slots of a subset R.

LEMMA 3.2. *For any matrix for a matching instance and any linear ordering of the set of all slots, there is exactly one ordered matching with that matrix.*

PROOF. Suppose A is a matrix and $\leq$ is a linear ordering of the set of all slots. It suffices to show that for each G and R, the pairs between G and R are uniquely determined. Suppose $A(G, R) = i$. Then the conditions (a) and (b) in the definition of the ordered matching and the values $A(G, X)$ for $X \in Red$ uniquely determine a set of i slots in G. These conditions and the values $A(X, R)$ for $X \in Green$ uniquely determine a set of i slots in R. Condition (c) then forces the i slots of G to be matched in order with the i slots of R.

Definition. If C is a subset of the set of slots of a given matching instance, a *matching on C* is a matching for the instance obtained by restricting the given instance to the slots of C (the preference lists are the same).

Definition. Suppose A and A' are two matrices for a given matching instance, their *Intersect Matrix I* is $I_{ij} = \min(A_{ij}, A_{ij}')$ where $i \in [1, m+1]$ and $j \in [1, n+1]$.

Definition. Given two matrices A and A' for a matching instance, a *Difference Set C of A and A'* is a subset of $\boldsymbol{S}$ in which, for any G_i ($i \in [1, m]$), the total number of its slots in C is the number of its slots in S subtracted by the sum of I_{ij}'s ($j \in [1, n+1]$) in the intersect matrix I of A and A'. Similarly for any R_j ($j \in [1, n]$), where I_{ij}'s are summed over $i \in [1, m+1]$.

Definition. Given two matrices A and A' for a given matching instance, an ordering $\leq$ of the slots, and a *Difference Set C of A and A'*, we define D and D' to be the unique ordered matchings on C with matrices A-I and A'-I, respectively, where I is the Intersect Matrix of A and A'.

THEOREM 3.1. *(1) Any slot in the given C has differently ranked partners in D and D'. (2) Suppose P is any matching on the complement of the given C with matrix I. Then* $P \cup D$ *has matrix A and* $P \cup D'$ *has matrix A'.*

PROOF. (1) Suppose not. Then A-I and A'-I have some item in common. That is impossible since it contradicts the definition of I. (2) Since the underlying sets of slots are disjoint, the matrix for $P \cup D$ is $(A\text{-}I)+I=A$. Likewise the matrix for $P \cup D'$ is A'.

Definition. Given two matrices A and A' for a given matching instance, an ordering $\leq$ of the slots, and a *Difference Set C of A and A'*, a *difference pair* is any pair which is in D or D'.

Hence, we define a difference pair of two *matchings* as a difference pair of the two *matrices* representing the two matchings. Likewise an intersect matrix of two *matchings*, a difference set of two *matchings*, etc, are defined.

Definition. Suppose a slot x has partner y and z in matchings M and M', respectively. $x \in X$, $y \in Y$ and $z \in Z$. The slot x *has a better partner in M than in M'* iff

$Y \prec_X Z$; it *has a partner at least as good in M as in M'* iff $Y \preceq_X Z$.

Definition. Suppose a slot x has partner y and z in ordered matchings M and M', respectively, and the linear ordering is $\leq$. The slot x *has a better partner in M than in M'* iff $y \leq z$ and $y \neq z$; it *has a partner at least as good in M as in M'* iff $y \leq z$.

Note, by (1) of Theorem 3.1, this definition and the previous one have no difference when applied to the slots in C, and therefore, both definitions are the same for the slots in C.

Definition. Given a matrix A and a pair (X, Y) where one of X and Y is a Green subset and the other is a Red subset, we define the *partial sum for (X, Y), $A(X, Y+)$*, as the number of slots of X which are matched to Y or to a partner that X prefers to Y in A. That is, $A(X, Y+) = \sum_{Y' \preceq_X Y} A(X, Y')$, where $Y' \preceq_X Y$ means X ranks Y' at least as good as Y.

LEMMA 3.3. *Given two matrices A and A' for a given matching instance, the following are equivalent:*

(1) For any G and any R, $A(G, R+) \geq A'(G, R+)$.

(2) All Green slots have at least as good partners in M as in M', where M and M' are the ordered matchings on S with A and A', respectively.

PROOF. (1) -> (2). Suppose (1) is true but (2) is false. That means, for some G, there is at least one of its slots having a worse partner in M than in M'. Suppose g is the first such slot in G according to the linear ordering $\leq$, and g's M partner belongs to R and g's M' partner belongs to R'. Then G prefers R' to R. Suppose $A'(G, R'+)$ is k. Then, g's position in G according to the linear ordering is within the first k positions. From (1), $A(G, R'+)$ is at least k. Since g is not paired in the first k pairs in M, there must be some slot in G which is after g in the linear ordering but paired in the first k pairs. This contradicts the condition that M is an ordered matching. Therefore, (2) must be true.

(2) -> (1). The partial sums being calculated are actually the numbers of best possible pairs for any Green subsets according to their preferences. From (2), since all slots from any G have at least as good partners in M as in M', then, for any R, $A(G, R+)$ must be no less than $A'(G, R+)$. Therefore, (1) is true.

THEOREM 3.2. *Given two matrices A and A' for a given matching instance, the following are equivalent:*

(1) All Green slots in the given C have better partners in D than in D'.

(2) For any G and any R, $A(G, R+) \geq A'(G, R+)$.

(3) All Green slots have at least as good partners in M as in M', where M and M' are the ordered matchings on S with A and A', respectively.

PROOF. Since D and D' are the ordered matchings on the given C with A-I and A'-I, respectively, then, by Lemma 3.3, all Green slots in C have at least as good partners in D as in D' iff (2) is true for A-I and A'-I. Since every slot in C has different partners in D and D', and (2) is true for A-I and A'-I is equivalent to (2) is true for A and A', then (1) is equivalent to (2). By Lemma 3.3, (3) is also equivalent to (2).

Note both definitions regarding the better partners are applicable to (1).

Definition. Given two matrices A and A' for a matching instance, A

dominates A' from Green's point of view iff any of the three conditions in Theorem 3.2 is true.

Definition. Given two matrices A and A' representing two matchings M and M' for a given matching instance, respectively, *M dominates M' from Green's point of view* iff A dominates A' from *Green*'s point of view.

Similarly, we can define the dominance relation *from Red's point of view.* Note the dominance defined here is a generalization of what is used in the standard stable matching problem.

COROLLARY 3.1. *The dominance relation is reflexive, transitive and antisymmetric.*

Definition. Given two matrices A and A' for a matching instance, *A strictly dominates A' from Green's point of view* iff A dominates A' from Green's point of view and $A \neq A'$.

Definition. A is *more popular than A'* iff the number of slots in the given C having *better* partners in D is greater than the number of slots in the same C having better partners in D'. If these two numbers are equal, then A and A' are *equally popular*. A matrix A is a *majority assignment* iff there is no matrix A' such that A' is *more popular than A*. A matrix A is a *strong majority assignment* iff A is *more popular than* all other matrices.

Similar to the dominance relation, we can define the popularity and majority relations of two matchings based on their representing matrices. Note the definition here is a generalization of Gärdenfors' notion.

Note also that the popularity relation is not transitive. One example is: in a given matching instance, there are one *Green* subset G with three slots, and nine *Red* subsets with one slot each. G's preference list is $R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8$, R_9 . All Red have the same preference: G. Suppose there are three matchings M, M' and M'', where $M = \{(g_1, r_4), (g_2, r_5), (g_3, r_6)\}$, $M' = \{(g_1, r_2), (g_2, r_3), (g_3, r_9)\}$, and $M'' = \{(g_1, r_1), (g_2, r_7), (g_3, r_8)\}$. Obviously M' s more popular than M, M'' is more popular than M', but M is more popular than M''.

B: An *objective matrix*, in which each element b_{ij} ($i \in [1,m]$, $j \in [1,n]$) is a non-negative real constant and represents the objective value of a matched pair (g, r) where $g \in G_i$, $r \in R_j$. Since this optimization is a maximization problem and the objective matrix is non-negative, a matched status is more preferred to a non-matched status.

objective function: a linear function of A_{ij} with the given objective matrix B, i.e., $\sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij} A_{ij}$.

Chapter 4. Algorithm I and Its Correctness Proof

Definition. (used in Algorithm I) For any G and any R which are acceptable to each other, R *is finished* with G, or G *has finished* with R, at any moment of the execution, iff at that moment, R is fully subscribed to G and possibly other Green's ranked higher than G by R. Hence, the complementary definitions, R *is not finished* with G, or G *has not finished* with R at any moment, mean either R is not provisionally assigned to G, or R is provisionally assigned to G, but still has replaceable slots for G, at that moment. Note that, if R is fully subscribed and its lowest ranked partner is better than G, then R will be dropped from G's preference list by Algorithm I. So, generally speaking, the statements, R *is not finished* with G, or G *has not finished* with R, mean R still has replaceable slots for G.

Definition. For a given matching instance, for simplicity, we will refer to the final matching generated by Algorithm I, with *Green* as proposers, as A_0. Similarly, if the roles of *Green* and *Red* in the algorithm are reversed, we will refer to the final matching generated by Algorithm I, with *Red* as proposers, as A_z.

Definition. For a given matching instance, we will refer to the final preference lists generated by Algorithm I, with *Green* as proposers, as the *Green-oriented Gale-Shapley lists*, or *GGS-lists* for short. Likewise, we obtain the *RGS-lists* if *Red* are proposers. Finally, if we take the intersection of its *GGS-list* and *RGS-list* by (1) for each Green or Red subset, dropping from its GGS-list those subsets which has already been removed in its RGS-list and (2) for each Red subset, marking the

G which is the worst partner of R in A_0 as finished on R's GGS-list, we obtain its *GS-lists*.

Algorithm I (*Green* as Proposers):

Provisionally assign each *Red* to a dummy slot, and each *Green* to be unsubscribed;
{Hence, each *Red* is not *finished* with any *Green*}

while (some G is undersubscribed) and (G's list contains an R not *finished* with G) do

begin
    R := first such *Red* on G's list;

*   Proposal and Replacement

    G_0 := the worst being provisionally assigned to R;
    repeat
        break a provisional assignment of R to G_0;
        provisionally assign this slot of R to G;
        G_0 := the worst being provisionally assigned to R
    until ($G = G_0$) or (G is fully subscribed)

*   Eliminate impossible pairings

    for each successor G_1 of G_0 on R's list do
        remove G_1 and R from each other's lists;
    if $G = G_0$, then
        mark R on G's list as *finished*;
end.

Some facts about the algorithm I:

(1) Initially, each subset of the Red is fully subscribed to the Green dummy slots of G_{n+1}, which can be replaced by any Green proposers.

(2) The Green must keep proposing until each subset is fully subscribed or every remaining Red on its preference list is *finished* with it.

(3) From (1) and (2), for any G and any R, if both G and R are acceptable to each other, then, upon the termination of the algorithm, either G is fully subscribed, or R has no partner ranked lower than G, or both.

(4) Since only higher-ranked new proposers can replace the existing partners of an R, the proposers ranked lower than R's existing partners can be removed from the related lists for the purpose of efficiency. This removal also guarantees that the new proposers to a subset of the Red must have higher ranks than its current worst partners and, therefore, at least one replacement can be made for each proposal.

(5) One iteration of the outer WHILE loop is one proposal by a subset G of the Green to a subset R of the Red. During this iteration, G takes all slots it can from R through replacements. Upon the end of this iteration, G and R are subscribed to each other and either G is fully subscribed, or R is *finished* with G, or both. Every iteration gives R, the proposed, a better partner, while keeping the partners of all other *Red* subsets unchanged.

(6) R is *finished* with G means G is the lowest partner of R. Note if R is fully subscribed to the Green's ranked higher than G (not including G), then R is removed

from G's list and we don't call this situation "*finished*". After an R is *finished* with a G, the number of the slots matched between the two can not increase, but can only stay the same or decrease due to possible replacements by any later proposers to R. An R, once being *finished* with a G, will either keep the *finished* status with the G (this does not mean there is no replacements going on), or be finally removed from G's list (because of a complete replacement), but will not return to the "*not finished*" status with G again. When a G proposes to some R on its list, all remaining Red's preceding R on the list must be *finished* with the G. That means, all Red's on G's list becoming *finished* with G follow the order in which G proposes -- the descending order of the preference.

(7) We claim that the maximum number of times that a G can propose to an R is no more than the number of slots of R. Suppose G always has free slots, either never used before, or regained sometime when G is replaced by other higher-ranked Green's. Whenever G has free slots, it should propose to the first Red on its list which is not *finished* with it. Suppose R is such a Red. While R is not *finished* with G, G can not free any its matched slots with R. Each time G proposes to R, R will give away at least one slot, previously matched with some lower-ranked partners of R, until R becomes *finished* with G. From (6), G can never propose to R again once R is *finished* with G. Thus the claim is true.

(8) We can prove by induction that, at the end of every iteration of the WHILE loop, if we exclude the free slots of those active G's (i.e. not fully subscribed and has some R not *finished* on its list), then the existing matching of the remaining

elements of the Green and the Red (let GR be its name) is stable. With their free slots being removed, those active G's become either fully subscribed or empty (if all of their slots are gone). Note that GR includes those G's that have free slots but every R on their lists is *finished* with them. Let GR_i be the existing matching of the GR after the i_th iteration. This assertion is obviously true with the first iteration of the loop when there is only one G that proposes to its first preference. After this first iteration, GR_1 has only one G and all R's and there is no possibility of a blocking pair. Suppose after the n_th iteration the loop is still not finished, and GR_n is a stable matching, we now prove that GR_{n+1} is also a stable matching. Suppose there is a blocking pair (g_b , r_b) in GR_{n+1} , $g_b \epsilon G_b$ and $r_b \epsilon R_b$.

Since $g_b \epsilon GR_{n+1}$, by the definition of GR_{n+1} , if g_b is free, then every R on G_b's list must be *finished* with G_b . That means, either R_b is removed from G_b's list or is *finished* with it. In both cases, R_b has been fully subscribed to the partners no worse than G_b . That indicates that r_b ranks its current partner at least as good as g_b , a contradiction to the definition of a blocking pair. Hence we conclude that g_b is not free in GR_{n+1} .

From (5), all R's in GR_{n+1} get the same or better partners as in GR_n . The fact that r_b prefers g_b to its partner in GR_{n+1} , implies that r_b must prefer g_b to its partner in GR_n . If g_b's partner in GR_n is the same as that in GR_{n+1} , we can easily conclude that (g_b , r_b) is a blocking pair in GR_n too, thus contradicting the assumption that GR_n is a stable matching. Note that it is impossible for g_b to have different partners from GR_n to GR_{n+1} if g_b is already matched in GR_n .

Now we consider the last case that g_b has a partner in GR_{n+1} but has no partners in GR_n. This situation indicates that G_b must be the proposer of the last iteration. At the end of this iteration either G_b is fully subscribed or the proposed must be *finished* with G_b. In either case, g_b's partner must be ranked no worse than the proposed. This fact, together with the definition of a blocking pair, implies R_b must be ranked higher than the proposed in G_b's list. Since G_b is the proposer of this iteration, R_b must have been fully subscribed to some G's ranked higher than G_b before. Therefore, r_b's partner has a higher rank than g_b, violating the definition of a blocking pair.

In summary, GR_{n+1} is also a stable matching, thus the condition remains true.

Correctness Proof:

(A) **Termination**:

a. The Termination of the Replacement Loop:

Since G and R are finite, and each pass of the loop decreases G's number of free slots by one, and replaces with G a slot of R provisionally assigned to some lower-ranked partner of R, G will finally become either fully subscribed or the worst partner of R, thus terminating the execution of the loop.

b. The Termination of the Outer Loop:

Suppose the cardinality of *Red* is m, and *Green* is partitioned into k non-overlapping subsets. From (7), the number of proposals a G can make to all R's on its list is no more than m. From (5), the maximum number of iterations of the outer loop is no more than mk.

(B) **Stability**:

If we consider the *Green* and the *Red* together as a set, *WGR*, then *WGR* is divided into two subsets, AG and *IGR*, after each iteration of the outer WHILE loop. The AG is the subset consisting of those active G's with their free slots, and the *IGR* the remaining elements of *WGR* with all R's, inactive G's, and those active G's without their free slots. From (8) above, the invariant of the outer WHILE loop is that the existing matchings of *IGR*'s (we called them *GR*'s there) are stable after every iteration. Before the loop starts, this invariant is obviously true since there is no pairing. Fact (8) proves that it remains true from iteration to iteration. When the loop terminates, the number of active G's is zero, hence $AG = \{\}$, and

$IGR = WGR$. The fact that GR is a stable matching implies that the existing matching on WGR is a stable matching too. This proves the stability of the final matching.

(C) **The Consequence of Nondeterminism**:

As stated at the beginning, the result of this algorithm is a matrix A where an element A_{ij} represents the number of slots matched between G_i and R_j. Two matchings are equivalent if for each G and each R, the number of the R's slots which are paired with the G's slots is the same for both matchings. The nondeterministic order of the proposals in the algorithm generates different paths before termination. However, it should not affect the result. That means, the matrix A should be the same for all sequences of the proposals. We will prove it with Newman's Confluence Lemma for Term Rewriting Theory [15, 19], which says that *local confluence implies confluence*. To apply this theorem, we must specify the following:

(a) A partially ordered set $(\{PM\}, <)$. The objects our algorithm works with are the set $\{PM\}$ which denotes the set of all existing partial matchings of the set WGR (the union of the Green and the Red, as stated in the proof of (B)) from every possible iterations of the outer WHILE loop until it is terminated. The partial order $<$ on this set is defined as follows: Given $PM_1, PM_2 \in \{PM\}$: $PM_1 < PM_2$ iff PM_2 dominates PM_1 from Red's point of view. From fact (5), the maximal elements in $(\{PM\}, <)$ are the matchings of WGR when the outer loop terminates.

(b) The propose-dispose reductions. Given a partial order $(X, <)$, a reduction is a map $f{:}X \to X$ such that for each $x \in X$, $x < f(x)$. That is, a reduction moves x one step down its path towards a maximal element. In $(\{PM\}, <)$, any iteration of the outer loop (the process of proposal and disposal) is such a reduction, which is obvious from fact (5).

A set of reductions is *confluent* iff for every x, any two paths from x to the bottom of the paths (i.e. when the loop terminates) end in the same maximal element. The set is *locally confluent* iff for every x, and any two reductions f_1 and f_2, there are two paths from $f_1(x)$ and $f_2(x)$ which end in the same (not necessarily maximal) element. Thanks to Newman's Lemma, we only need to prove that the above propose-dispose reductions are locally confluent.

Suppose PM_0 is an element of $\{PM\}$. If there is less than two active G's in PM_0, then we do not need to prove the confluence. Now suppose G_1 and G_2 are any two active G's in PM_0. The reduction $f_1=(G_1, R_1)$ means G_1 proposes to R_1, the first R on its list not finished with it in PM_0, and generates a new matching PM_1. Similarly, $f_2=(G_2, R_2)$ means G_2 proposes to R_2, the first R on its list not finished with it in PM_0, and generates PM_2. Note that, although f_1 and f_2 seem to lose generality, they meet the definition of the local confluence since the choices of PM_0, G_1 and G_2 are arbitrary.

Suppose $R_1=R_2=R_0$. Without loss of generality, we assume that R_0 ranks G_1 higher than G_2. If R_0 is no longer on G_2's list in PM_1, which indicates G_2 and R_0 are impossible pairing in the presence of G_1, then we claim applying $f_1=(G_1, R_0)$ on PM_2 will generate PM_1. In other words, applying f_1 and f_2 to PM_0 in the sequence (f_2, f_1) will generate the same results as applying only f_1 to PM_0. This is true since

(1) In both paths, when f_1 is executed, the number of slots G_1 can replace is only determined by the number of free slots that G_1 has and the number of replaceable slots that R_0 has for G_1.

(2) One reduction only increases the number of slots matched between the proposer and the proposed, and reduces the number of slots matched between the proposed and those G's being replaced, leaving all others unchanged. Therefore, in the following proof, we will not discuss other R's than this R_0 since they are not effected by this reduction, and, for the same reason, we will only be concerned with those G's which are R_0's partners in PM_0 and are ranked lower than G_2 or G_1 by R_0.

From (2), in the path (f_2, f_1), the execution of f_2 can not change the number of free slots that G_1 has. Also, since R_0 ranks G_1 higher than G_2, f_2 does not change the number of replaceable slots that R_0 has for G_1 either. So, the number of slots matched between G_1 and R_0 in both paths should be the same. Each of those G's being fully or partially replaced by either G_1 or G_2 regains the same amount of free slots, since for both paths (a) the replacements always start from the lowest-ranked partner, (b) the total numbers of replacements are the same, and (c) the final numbers of G_2's slots matched with R_0 are the same($=0$). Therefore, the results of the sequence (f_2, f_1) should be the same as f_1 only.

Now if R_0 is still on G_2's list in PM_1, we claim that applying $f_1=(G_1, R_1)$ on PM_2 and applying $f_2=(G_2, R_2)$ on PM_1 will come to the same matching PM_3. This claim means, under the circumstance, applying f_1 and f_2 to PM_0 in two different sequences will generate the same results. That R_0 is still on G_2's list in PM_1

indicates that G_1 is full in PM_1 and R_0 still has some replaceable slots for G_2 after PM_1. Following all the facts, we can assume that, in PM_0, R_0 has n replaceable slots for G_1 and m replaceable slots for G_2, and G_1, G_2 has k and q free slots, respectively, and $n \geq m > k$. In the path (f_1, f_2), G_1 takes R_0's k slots, previously matched to its lowest partners, and becomes fully subscribed, then G_2 has min(q, m-k) slots from R_0's remaining m-k slots. In the path (f_2, f_1), G_2 takes R_0's min(q, m) slots, previously matched to its lowest partners, leaving m-min(q, m) slots still replaceable by G_2, then G_1 comes and has k slots matched with R_0, which consists of min(k, m-$min(q, m)$) slots from those still available to G_2 and k-$min(k, m$-$min(q, m))$ from G_2, leaving $min(q, m)$-$(k$-$min(k, m$-$min(q, m)))$ slots to G_2. This expression is equal to $min(q, m$-$k)$ (to verify it, check the cases $q > m$ and $q \leq m$). This proves that the numbers of slots matched between G_1 and R_0, or between G_2 and R_0 are the same for both paths. Following the same reasoning above, we can find the numbers of free slots regained by each of the replaced G's, are also the same for both paths.

Now if R_1 is not R_2, we need to consider whether G_1 has replaced some slots of G_2 in PM_1, or G_2 replaces some slots of G_1 in PM_2, or both. If G_1 has not replaced any slots of G_2 in PM_1, and G_2 has not replaced any slots of G_1 in PM_2, then the numbers of free slots of G_1 and G_2 are the same for the paths (f_1, f_2) and (f_2, f_1) and we claim that applying $f_1 = (G_1, R_1)$ on PM_2 and applying $f_2 = (G_2, R_2)$ on PM_1 will generate the same matching PM_3. However, if one of these situations does happen, then the numbers of free slots of G_1 and G_2 will be different for different paths. Furthermore, if the proposed has enough slots available to the proposer with more free slots, then the results will be quite different for the above paths. The following method will solve this problem:

(1) If G_1 has regained some free slots in PM_2 and R_1 still has free slots in PM_1, apply (f_2, f_1) on PM_1; otherwise, apply f_2 on PM_1;

(2) If G_2 has regained some free slots in PM_1 and R_2 still has free slots in PM_2, apply (f_1, f_2) on PM_2; otherwise, apply f_1 on PM_2.

For example, if G_2 has regained k free slots in PM_1, and R_2 still has q free slots in PM_2, then applying f_2 on PM_1 means G_2 can get $\min(k, q)$ more slots matched with R_2 than in PM_2. Applying (f_1, f_2) on PM_2 gives G_2 k free slots and $\min(k, q)$ more matches with R_2 than in PM_2.

This proves that the above propose-dispose reductions are locally confluent. By Newman's Lemma, they are confluent.

Chapter 5. Pareto Efficiency

Pareto Efficiency considerations can be used to improve the result of a stable assignment when a weak order is introduced into the preference lists. A list is *linear* iff there are only strict preferences. A list is in *a quasi-linear* (or *weak*) order iff there are ties in the linear order. A matching M is *Pareto Efficient* iff for any pairs (g_1, r_1) and (g_2, r_2) in M, if r_1 ranks g_1 and g_2 the same, and r_2 ranks g_1 and g_2 the same, then not both (1) g_1 ranks r_2 at least as good as r_1, and (2) g_2 prefers r_1 to r_2. It must also hold from the other perspective, where g and r are interchanged.

Note that the pairing (g_1, r_1) and (g_2, r_2) in M is a stable one by definition since r_1 and r_2 rank g_1 and g_2 the same. However, as pointed out by Gärdenfors, this pairing may not be the best in terms of the satisfaction of the parties involved. Obviously, if (1) and (2) are both true, then we can improve the satisfaction of g_2 without sacrificing stability and without affecting anybody else's satisfaction by forming a new stable pairing (g_1, r_2) and (g_2, r_1).

As suggested by Gärdenfors [4], if we allow the quasi-linear order in the preference lists, extra work is needed to check each stable matching and make it Pareto efficient. In the preference lists used by our approach, we only allow a strict linear order to be specified between subsets of the two sets. Although a quasi-linear order relationship exists between the slots of one set and the slots of the other set, Pareto Efficiency is guaranteed. If slots g_1 and g_2 are ranked the same by slot r_1 or r_2, then they must be in the same subset of the Green and have the same preference over r_1 and r_2 . This guarantees that (1) and (2) can not be both true.

Chapter 6. Majority Assignments

According to the theory of voting, the Majority Assignment property is a very desirable feature for any assignment. This is especially true when stability is a requirement. Gärdenfors proved that in the standard matching problem, all stable matchings are majority assignments, and, therefore, equally popular. Unfortunately, it is possible when weak orders are allowed that a stable matching, or even a stable and Pareto Efficient matching, is not a majority assignment. For example, in the following instance by Gärdenfors [4], we have

$$\begin{array}{llll} a_1: & b_1 & b_2 & b_3 \\ a_2: & b_3 & b_1 & b_2 \\ a_3: & b_3 & b_2 & b_1 \end{array} \qquad \begin{array}{llll} b_1: & (a_3 & a_1) & a_2 \\ b_2: & a_2 & a_1 & a_3 \\ b_3: & a_2 & a_1 & a_3 \end{array}$$

where () means tie. Then both matchings M: $\{(a_1, b_1), (a_2, b_3), (a_3, b_2)\}$ and M': $\{(a_1, b_2), (a_2, b_3), (a_3, b_1)\}$ are stable and Pareto Efficient but M is more popular than M'.

However, Gärdenfors [4] proved that, if there exist majority assignments in his case, at least one of them will be stable (Theorem 6.1). Here we prove that in our version of the stable matching problem any stable matching is a majority assignment.

THEOREM 6.1. *Any strong majority assignment is stable.*

Proved by Gärdenfors for his general cases.

THEOREM 6.2. *Any stable matching is a majority assignment.*

PROOF. Suppose M is a stable matching but not a majority assignment.

That means there is at least one matching M' that is *more popular than M*. Let C be the difference set of M and M' as defined above.

(1) If D' has no pair that has any slot from G_{m+1} or R_{n+1}, then, obviously the cardinality of the set C must be an even number, say, $2k$ $(k \geq 1)$. Since M' is preferred by a majority to M, there are at least two more slots in C which prefer M' to M. Since there are only k pairs in D', there is at least one pair matched in M' who both prefer M' to M. According to the definition of D', they are not matched in M. Hence they are a blocking pair of M. Therefore, M is not stable. This contradicts the assumption.

(2) If D' has some pairs with slots from G_{m+1} or R_{n+1}, then, we can divide C into two subsets: one, C_1, with the slots that are unmatched in M', and the other, C_2, with the slots that are matched in M'. For every slot in C_1, M' is at most as good as M since they are unmatched in M'. Since M' is preferred to M, C_2 (which also has an even cardinality) must have at least two more slots which prefer M' to M. This theorem is proved by the same argument as in (1).

COROLLARY 6.1. *Two stable matchings are equally popular.*

THEOREM 6.3. *Let M and M' be stable matchings, and suppose (g, r) is one of its difference pairs. Then one of g and r prefers M to M', and the other prefers M' to M.*

PROOF. Let C be the difference set of M and M'. Let X and Y (respectively X' and Y') denote the sets of Green and Red slots in C which would prefer M to M' (respectively M' to M). In M, there can be no pair (g, r) with $g \in X$

and $r \in Y$, for such a pair would block M'. So every man in X has an M-partner in Y', and therefore $|X| \leq |Y'|$. Based on the similar reasoning, $|X'| \leq |Y|$. But $|X| + |X'| = |Y| + |Y'|$, since the left-hand side is the number of Green slots, and the right-hand side the number of Red slots. It follows that $|X'| = |Y|$ and $|X| = |Y'|$ which proves the theorem. Furthermore, the fact $|X| + |Y| = |X'| + |Y'|$ also proves Corollary 6.1.

COROLLARY 6.2: *If A strictly dominates A' from Green's point of view, then, for any G, we have $worst(G, A) \preceq_G worst(G, A')$. Furthermore, if, in addition, $I = min(A, A') = 0$, then $worst(G, A) \prec_G worst(G, A')$.*

PROOF. By the definition of dominance, the first assertion is clear. For the second part, since no element is nonzero in both A and A', every pair of A and A' is a difference pair of A and A'. By Theorem 6.3, for every difference pair in A and A', one slot prefers A and the other prefers A'. Since each slot in a difference pair gets differently ranked partners in A and A', then one slot in the pair gets a higher ranked partner in A than in A', and the other gets a lower ranked partner in A than in A'. Since A strictly dominates A', then, for any G, all of G's slots get higher ranked partners in A than in A'. Therefore, worst$(G, A) \prec_G$ worst(G, A').

Before finishing this section, we want to emphasize again the significance of the ordered matchings D and D' in the correct definition of majority assignments. In the following example, we show that, without the ordered matchings D and D', stability does not guarantee a majority assignment.

$G_1(4)$:	R_1	R_2	R_3	$R_1(2)$:	G_2	G_1
$G_2(1)$:	R_3	R_1	R_2	$R_2(2)$:	G_2	G_1
				$R_3(1)$:	G_1	G_2

where (n) means there are n slots in the subset. We assume the slots are indexed in increasing order.

Here we have two stable matchings, M: $\{(g_{11}, r_{11}), (g_{12}, r_{12}), (g_{13}, r_{21}), (g_{14}, r_{22}), (g_{21}, r_{31})\}$, and M': $\{(g_{11}, r_{12}), (g_{12}, r_{21}), (g_{13}, r_{22}), (g_{14}, r_{31}), (g_{21}, r_{11})\}$. Both of them are ordered. Then M and M' do not have an equal number of slot voters since g_{12}, g_{14}, g_{21} prefer M to M', r_{11}, r_{31} prefer M' to M, and g_{11}, g_{13}, r_{12}, r_{21}, r_{22} have partners with the same rank in M and M'. Let C be $\{g_{11}, g_{21}, r_{11}, r_{31}\}$. Then the slot voters in D: $\{(g_{11}, r_{11}), (g_{21}, r_{31})\}$ and D': $\{(g_{11}, r_{31}), (g_{21}, r_{11})\}$ are equally divided. This definition also applies to the standard stable matching problem. In both cases, when $I = 0$, C is the whole set of slots, and D and D' are M and M' themselves.

Chapter 7. Green-Optimality and Red-Optimality

Suppose we apply Algorithm I (*Green* as proposers) to an instance and get a stable matching M_0. M_0 is **Green-Optimal** means that, for any stable matching M of the instance ($M \neq M_0$), all Green slots in the C set of M_0 and M must prefer M_0 to M. In other words, M_0 dominates all stable matchings for the instance from Green's point of view. Similarly, if we apply the Algorithm I (*Red* as proposers) to an instance and get a stable matching M_z. That M_z is **Red-Optimal** means, for any stable matching M of the instance ($M \neq M_z$), all Red slots in the C set of M_z and M must prefer M_z to M. That is, M_z dominates all stable matchings for the instance from Red's point of view.

THEOREM 7.1. *M_0 is Green optimal.*

PROOF. Suppose M_0 is not Green optimal. That means there is at least a stable matching M and at least a Green slot, g, in the C set of M and M_0 that prefers its M-partner, r, to its M_0-partner, r_0. Suppose $g \epsilon G$, $r \epsilon R$ and $r_0 \epsilon R'$. Since $g \epsilon C$, by the definition of C, $r \epsilon C$, $r_0 \epsilon C$ and $R \neq R'$. Then R must be in front of R' in G' preference list. That means R must have rejected G's proposal during the execution of Algorithm I for M_0. Suppose, without loss of generality, this is the first occasion during the execution that a Red rejected a proposal, and suppose that this rejection took place because of the engagement of r to g' ($g' \epsilon C$ and r prefers g' to g). Then g' can not prefer any other Red to r (for no Red had previously rejected a proposal). So g' prefers r to its M-partner. Then M is blocked by (g', r). This is a contradiction to the assumption that M is stable.

The same proof proves M_z is Red-Optimal.

THEOREM 7.2. *In M_0, each R_j ($j \in [1,n]$) has the worst partners that it can have in any stable matching.*

PROOF. Suppose not. That means there is at least one stable matching M and at least one Red slot, r, in the C set of M and M_0 that prefers its M_0-partner, g_0, to its M-partner, g. According to Theorem 7.1, g_0 must prefer r to its M-partner. Since $(g_0, r) \in M_0$ is a difference pair of M_0 and M, it is not in M. Therefore, (g_0, r) is a blocking pair to M.

LEMMA 7.1. *In a stable matching instance that allows unacceptable partners and in which the sets of Green and Red are of unequal size, all the slots of the Green and the Red are each partitioned into two sets - those that have partners in all stable matchings and those that never have partners.*

Gusfield and Irving proved this in their book. By this Lemma, we only need to consider the case where the Green and the Red have the same number of slots.

THEOREM 7.3. *For a given stable matching instance, there is no matching, stable or otherwise, in which every Green slot has a partner which it strictly prefers to its M_0-partner.*

PROOF. Suppose M is any matching such that $M \neq M_0$. Then the C set of M_0 and M is not empty. Theorem 7.1 shows every Green slot in C prefers its M_0-partner to its M-partner, if M is a stable matching. So M can not be a stable matching. Suppose M is an unstable matching with this property. We first show that

Theorem 7.3 is true for the instance in which the total number of slots for Green is equal to that for Red. Now consider the termination of a particular execution of Algorithm I (with Green as the proposer). When the algorithm terminates the proposal accepted by the Red slot r will be its first. That means no Green slot is rejected by r. According to the definition of M, r must have different partners in M and M_0. If r's partner in M is g, then g must prefer r to its partner in M_0, so r must have rejected g. This is a contradiction. By Lemma 7.1, we do not need to prove other cases.

Let A_0 and A_z represent M_0 and M_z respectively, then we can put all of these theorems in matrix form.

Chapter 8. Structure of the Solution Space

Gusfield and Irving [8] proved that the solution space of a stable matching instance with strict preferences is a distributive lattice. This is not true for the solution space in the general case that Gärdenfors [4] proposed, since he claimed the Green-optimal elements may not be unique in a situation which contains some non-linear preference orders. However, it remains true for the solution space in our group preference case.

A **distributive lattice** is a partial order $(S, \leq)$ in which

i. each pair of elements $a \in S$, $b \in S$ has a greatest lower bound, or *meet*, denoted by $a \wedge b$, so that $a \wedge b \in S$, $a \wedge b \leq a$, $a \wedge b \leq b$, and there is no element $c \in S$ such that $c \leq a$, $c \leq b$ and $a \wedge b < c$;

ii. each pair of elements $a \in S$, $b \in S$ has a least upper bound, or *join*, denoted by $a \vee b$, so that $a \vee b \in S$, $a \leq a \vee b$, $b \leq a \vee b$, and there is no element $c \in S$ such that $a \leq c$, $b \leq c$ and $c < a \vee b$;

iii. the distributive laws hold, namely $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ and $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$.

Let $\mathcal{A}$ be the solution space, the set of all stable matrices, for a given stable matching instance. Define $\leq$ to be the partial order on $\mathcal{A}$ which satisfies the following: for any A, $A' \in \mathcal{A}$, $A \leq A'$ iff A dominates A' from Green's point of view. Define $<$ by $A < A'$ iff $A \leq A'$ and $A \neq A'$. We want to prove that $(\mathcal{A}, \leq)$ as defined is a distributive lattice.

Note we have three definitions for the Green-domination relation. For convenience, we use the definition that M dominates M' from Green's point of view iff all *Green* slots have at least as good partners in M_1 as in M_1', where M_1 and M_1' are the unique ordered matchings with A and A', respectively. The following theorems are easy to prove by the methods similar to those used by Gusfield and Irving in their standard matching case:

LEMMA 8.1. *Given two distinct stable matrices A and A' for a given matching instance, let M and M' be the unique ordered matching with A and A', respectively. If each Green slot is given the better of its partners in M and M', then the result is a stable ordered matching.*

PROOF. (1) First we prove the result is a matching. Suppose it is not. Then we must have at least two Green slots, g and g', that receive the same partner r. Suppose $(g, r) \in M$ and $(g' r) \in M'$. This means g prefers M to M' and g' prefers M' to M. Applying Theorem 6.3 to (g, r), we have r prefers M' to M. Applying Theorem 6.3 to (g', r), we have r prefers M to M'. This is a contradiction. Therefore, the result is a matching.

(2) We prove the result is stable. Suppose it is not. Then we have at least one blocking pair (g, r) in it. By the definition of the blocking pair, r must strictly prefer g to its partner and g must strictly prefer r to both its partners in M and M' since g receives the better of its partners in M and M' as its partner in the new matching. Since r's partner in the new matching is either its M-partner or its M'-partner, then either M or M' will be blocked by (g, r). This is a contradiction to the

condition that both M and M' are stable matchings.

(3) We prove the result is ordered. Suppose it is not. Then we have (g_1, r_1) and (g_2, r_2) in the result, such that $g_1 \leq g_2$ and $r_2 \leq r_1$. By definition every pair in the result is from either M or M'. If both (g_1, r_1) and (g_2, r_2) belong to M or M', then either M or M' is not ordered. Otherwise, suppose $(g_1, r_1) \epsilon M$, $(g_2, r_2) \epsilon M'$ and $(g_1, r_1') \epsilon M'$. Since (g_1, r_1) is in the result, by definition, we have $r_1 \leq r_1'$. That means $r_2 \leq r_1 \leq r_1'$. Then (g_1, r_1') and (g_2, r_2) in M' implies M' is not ordered which contradicts the condition that both M and M' are ordered. Therefore, the result is an ordered matching.

LEMMA 8.2. *Given two distinct stable matrices A and A' for a given matching instance, let M and M' be the unique ordered matching with A and A', respectively. If each Green slot is given the poorer of its partners in M and M', then the result is a stable ordered matching.*

PROOF. By Theorem 6.3, if each Green slot is given the poorer of its partners in M and M', then each Red slot receives the better of its partners in M and M'. So this lemma is actually the dual of Lemma 8.1.

We denote by $A \wedge A'$ the stable matrix that represents the stable ordered matching in which each Green slot obtains the better of its partners in M and M', and by $A \vee A'$ the stable matrix that represents the stable ordered matching in which each Green slot receives the poorer of its partners in M and M', where M and M' are the unique ordered matching of A and A', respectively.

THEOREM 8.1. *For a given matching instance, the partial order $(A, \leq)$*

forms a distributive lattice, with $A \wedge A'$ representing the meet of A and A', and $A \vee A'$ the join.

PROOF. By Lemma 8.1, $A \wedge A' \in \mathcal{A}$. By Theorem 3.2, $A \wedge A' \leq A$, $A \wedge A' \leq A'$. If there is an $A^* \in \mathcal{A}$ such that $A^* \leq A$ and $A^* \leq A'$, then, by the definition of ≤ and Theorem 3.2, all Green slots have at least as good partners in M^* as in M and M', where M^*, M and M' are the ordered matchings with A^*, A and A', respectively. That is, the M^* partner of each Green slot is at least as good as the better of its partners in M and M'. Then $M^* \leq M \wedge M'$ by the definition of ≤. So, by definition of ≤, $A^* \leq A \wedge A'$ and $A \wedge A'$ is the greatest lower bound (meet) for A and A'. The proof that $A \vee A'$ is the least upper bound (join) is similar. Therefore $(\mathcal{A}, \leq)$ is a lattice.

Next we prove that the first distributive law is satisfied. Let X, Y, Z be any stable matrices. Let $H = X \wedge (Y \vee Z)$ and $J = (X \wedge Y) \vee (X \wedge Z)$. We want to prove $H=J$ by showing that every Green slot g has the same partner in M_H as it has in M_J, where M_H and M_J are the unique stable ordered matchings with H and J, respectively. Suppose M_X, M_Y and M_Z are the unique stable ordered matchings with matrices X, Y and Z, respectively. If g's M_Y-partner r is the same as its M_Z-partner, then g is partnered by whichever of r and its M_X-partner it most prefers. Otherwise, it's easy to verify that, in both M_H and M_J, g is partnered by its M_Z-partner if g prefers M_Y to M_Z to M_X, by its M_Y-partner if g prefers M_Z to M_Y to M_X, and in all other cases by its M_X-partner. Therefore, $H=J$.

The second distributive law is proved in a similar way.

THEOREM 8.2: *Given two distinct stable matrices A and A' for a given matching instance, the following three definitions for matrix E are equivalent:*

(1) Let M and M' be the unique ordered stable matching with A and A', respectively. Then E represents the ordered stable matching M_E where each Green slot is given the better of its partners in M and M'.

(2) For any G and any R, E(G, R+) = max (A(G, R+), A'(G, R+)).

(3) Let matrix H represent the ordered stable matching on a difference set C of A and A', in which each Green slot is given the better of its partners in the matchings D and D'. Then E is H+I, where I is the intersect matrix of A and A'.

PROOF. (a) We prove (1) and (2) are equivalent.

=>: Suppose (1) holds. From (1), E dominates *A* and *A'* from Green's point of view. Then, by Theorem 3.2, for any G and any *R*, *E(G, R+)* ≥ max (*A(G, R+), A'(G, R+)*). Suppose for some *G* and *R, E(G, R+)* > max (*A(G, R+), A'(G, R+)*). Suppose *R* is the first such Red on *G*'s preference list. Since the partial sums are the numbers of pairs matched in the specified initial segments of the preference lists, then, at least one Green slot from *G* is paired in M_E with a Red slot which is ranked higher than its partners in *M* and *M'*. This is a contradiction to the rule specified in (1) about the construction of M_E . Therefore, only the equality sign holds.

<=: Suppose (2) holds. From (2), *E* dominates *A* and *A'* from Green's point of view. Then, by Theorem 3.2, all Green slots have at least as good partners in M_E as in *M* and *M'*. Suppose there is a Green slot from some *G* that ranks its M_E

partner better than any of its partners in M and M'. Suppose R is the first such Red on a G's preference list. Since no Green slots have worse partners in M_E than in M and M', $E(G, R+) > \max(A(G, R+), A'(G, R+))$ which is a contradiction to (2). Therefore, (1) is true. In summary, (1) and (2) are equivalent.

(b) We use (a) to prove (3) is equivalent to (2). It is easy to show that if A and A' are stable, then $A\text{-}I$ and $A'\text{-}I$ must be stable too. By substituting D for M, D' for M', H for E, $A\text{-}I$ for A, and $A'\text{-}I$ for A', in (1) and (2), we have

(1)' Let D and D' be the unique ordered stable matching with $A\text{-}I$ and $A'\text{-}I$, respectively. Then H represents the ordered stable matching M_H where each Green slot is given the better of its partners in D and D'.

(2)' For any G and any R, $H(G, R+) = \max(\{A\text{-}I\}(G, R+), \{A'\text{-}I\}(G, R+))$. That is, each of the partial sums of H_{ij}'s starting from the top of the preference list of any G_i is equal to the greater of that in $A\text{-}I$ and $A'\text{-}I$.

By using the result of (a), H as defined in (1)' is the same as it is defined in (2)'. By adding matrix I to the H as defined in (1)', we can get E as defined in (3). By adding the same I to each of $H, A\text{-}I, A'\text{-}I$ in (2)', we can get E as defined in (2), since $E=H+I$, $A=(A\text{-}I)+I$, $A'=(A'\text{-}I)+I$, and adding I to each of $H, A\text{-}I, A'\text{-}I$ does not change the relation in (2)'. Thus, E as defined in (3) is the same as that defined in (2). Therefore, all three definitions are equivalent.

Obviously E is the meet of A and A'. Similarly, for the join of A and A', we have the following theorem:

THEOREM 8.3: *Given two distinct stable matrices A and A' for a given* *matching instance, the following three definitions for matrix F are equivalent:*

(1) Let M and M' be the unique ordered stable matching with A and A', respectively. Then F represents the ordered stable matching M_F where each Green slot is given the poorer of its partners in M and M'.

(2) For any G and any R, F(G, R+) = min (A(G, R+), A'(G, R+)).

(3) Let matrix F' represent the matching on a difference set C of A and A', in which each Green slot is given the poorer of its partners in the matchings D and D'. Then F is F'+I, where I is the intersect matrix of A and A'.

Given two stable matrices for a matching instance, the algorithm below calculates the meet and join of the matrices. Suppose A and A' are the given matrices, E is their meet and F is their join.

```
E := 0;  F := 0;   { initialize two matrices }
for i := 1 to m do  { m, n are the partitions of Green and Red }
begin
        Suppose B_1 , B_2 , ..., B_{n+1} and B'_1 , B'_2 , ..., B'_{n+1} are A_ij's and
        A'_ij's sorted in the descending order of G_i's preference over R_j's,
        respectively;
        s := 0; s' := 0; e := 0; f := 0;  { the partial sums for A, A', E, F }
        for t := 1 to n+1 do
        begin
                s := s + B_t;  s' := s' + B'_t;
                x := max(s, s');        y := min(s, s');
                E_{i,t} := x - e;          F_{i,t} := y - f;
                e := x;          f := y;
        end;
end;
for i := 1 to n do  { for the row m+1 - unmatched Red}
for j := 1 to m do
begin
        E_{m+1,i} := E_{m+1,i} + (A_ij - E_ij);
        F_{m+1,i} := F_{m+1,i} + (A_ij - F_ij);
end.
```

Chapter 9. Representation of All Solutions (Algorithm II)

We have proved that the solution space of a group stable matching instance is a finite distributive lattice. Algorithm I guarantees there is at least one stable matching for any given matching instance. If A_0=A_z , there is only one solution for the given instance. However, it is common to have more solutions. In particular, Knuth [16], Irving and Leather [11], Hwang [9] pointed out that, although finite, the number of stable matchings can grow exponentially with problem size in the standard stable matching instance. Since our group stable matching problem includes the standard case (when every subset has only one slot), the size of its solution lattice can also grow exponentially with problem size. To construct an efficient algorithm to optimize the objective function over the solution space, we need to find a good representation of all elements in the solution lattice.

Birkhoff's representation theorem [6] shows that every finite distributive lattice is isomorphic to a ring of sets. However, as Gusfield and Irving [8] point out, from an algorithmic application point of view, Birkhoff's theorem is not a feasible tool for developing the structure of stable matchings, since we must first know the solution lattice before we can use the theorem. The following approach generalizes what Gusfield and Irving [8] have done with the standard matching problem by introducing some new concepts and theorems which are specific to the grouped stable matching instances.

Definition. A matching is a *full* matching if, for any $G \in Green$ and any

$R \epsilon Red$, none of its slots are unmatched.

By Lemma 7.1, we can drop those unmatched slots in the following discussion since from now on we only discuss stable matchings and stable matrices. After dropping all unmatched slots, all pairs with dummy slots are not considered and matrices are redefined as mxn matrices without G_{m+1} and R_{n+1}. In addition, all R's become fully subscribed and some R's may change their status with some G from '*not finished*' to '*finished*' because G now becomes the lowest ranked partner of R. In the following discussion, all stable matchings are full matchings.

Here we also need to take a new look at the definition of the GS-lists. Previously, we defined GS-lists based on the applications of Algorithm I on the original preference lists. We will refer to them as the *initial GS-lists*. Now we try to define the GS-lists more generally so they can be used and updated for every stable matching. Before doing that we need some new definitions.

Definition. For a full matching A, let X and Y be a pair of Green and Red subsets. Then $AV(A, X, Y)$ is the number of slots in X that are available to Y in A, i.e., the total number of X's slots that are paired with some subset ranked lower than Y by X.

Obviously, $AV(A, G, R) = \sum_{R <_G R'} A(G,R')$ and $AV(A, R, G) = \sum_{G <_R G'} A(G',R)$.

Definition. For a full stable matching A, the *GS-lists for A* are obtained from the original preference lists by (1) removing a pair (G, R) from the lists if $A(G, R) = 0$ and $AV(A, R, G) = 0$; and (2) marking R as *finished* in G's list if $A(G, R) \neq$

0 and $AV(A, R, G) = 0$; marking G as *finished* in R's list if $A(G, R) \neq 0$ and $AV(A, G, R) = 0$.

Since the pair (G, R) is removed from the GS-lists for A if $A(G, R) = 0$ and $AV(A, R, G) = 0$, then R is *not finished* with G *on G's GS-list for A* implies $AV(A, R, G) \neq 0$.

Definition. Let A be a full stable matrix, G a Green subset and R a Red subset. R is *available* to G in A iff $AV(A, R, G) \neq 0$.

Recall that if $A_z(G, R) = 0$ and $AV(A_z, G, R) = 0$, then the pair (G, R) is dropped from the RGS-lists by Algorithm I, since the R is worse than G's worst partner in A_z. For any stable matching A that dominates A_z from Green's point of view, worst(G, A) can not be worse than worst(G, A_z), so the pair (G, R) can be removed from the GS-lists for any A to improve the efficiency of other related algorithms. Therefore, for each G and each R, we make its initial GS-list as the intersection of its GGS-list and its RGS-list and use it as the GS-list for A_0.

LEMMA 9.1. *For each Green subset and each Red subset, its initial GS-list is the intersection of its GS-list for A_0 and its GS-list for A_z.*

PROOF. The intersection is the same operation as we defined before. It suffices to prove that the preference lists generated by Algorithm I (Green as proposer) are the GS-lists for A_0 defined above. For each pair (G, R) eliminated by Algorithm I, it is obvious that $A(G, R) = 0$ and $AV(A_0, R, G) = 0$ since R ranks G lower than R's worst partner, and will only accepts better partners later. For each R marked as finished on G's list by Algorithm I, it is G's lowest ranked partner, and, therefore, $A(G, R) \neq 0$ and $AV(A_0, G, R) = 0$. On the other hand, for any pair (G, R), if $A(G, R) = 0$ and $AV(A_0, R, G) = 0$, R must rank its worst partner higher than G,and (G, R) must be eliminated during the process of Algorithm I. If $A(G, R) \neq 0$ and $AV(A_0, G, R) = 0$, then R is G's lowest ranked partner and Algorithm I will mark R as finished on G's list. So, the preference lists generated by Algorithm I (Green as proposer) are the GS-lists for A_0 defined above.

By the definition of stability, it is easy to prove the following lemma:

LEMMA 9.2. *For any G in a stable matrix A, the GS-list of G for A must satisfy one of the following conditions:*

(1) every R on the list is finished with G; or

(2) there is an R' on the list such that every Red before R' on the list, if any, must be finished with G, and R' and every Red after it is available to G. In addition, if R is any Red after R', then $A(G, R) = 0$ while $A(G, R')$ can be either zero or not.

LEMMA 9.3. *Let A and A' be two stable matrices such that A strictly dominates A' from Green's point of view. Suppose $G = worst(R, A)$. Then $A(G, R) \geq A'(G, R)$.*

PROOF. We have A' strictly dominates A from Red's point of view since A strictly dominates A' from Green's point of view. Then, worst(R, A') $\leq_R$ worst(R, A) (see Corollary 6.2). Let G' = worst(R, A'). So $G' \leq_R G$. That means G' is not after G in R's GS-lists for A and for A'. If G' is before G,then $A'(G, R) = 0 < A(G, R)$. If G' is G, since A and A' are full matchings and G and G' are the worst partners in each, then $A(R, G+) = A'(R, G+)$. If there is no Green before G in R's GS-lists, then $A(G, R) = A(R, G+) = A'(R, G+) = A'(G, R)$. Otherwise, let G'' be the last Green before G in R's GS-lists. Since A' strictly dominates A from Red's point of view, we have $A(R, G''+) \leq A'(R, G''+)$ (see Theorem 3.2). Then $A(G, R) = A(R, G+) - A(R, G''+) \geq A'(R, G+) - A'(R, G''+) = A'(G, R)$.

In other words, Lemma 9.3 is true if R is finished with G on G's GS-list for A.

COROLLARY 9.1. *Let A and A' be two stable matrices and A' strictly dominates A from Red's point of view. Suppose R = worst(G, A'). Then $A'(G, R) \geq A(G, R)$.*

COROLLARY 9.2. *Let $\mathbf{A}$ be the solution space for a given stable matching instance, A_z be the Red-Optimal matrix, and $A \in \mathbf{A}$ be any stable matrix other than A_z. Let G be any Green subset whose GS-list ends with an R that is finished with G in A. Then, for all $R \in Red$, $A(G, R) = A_z(G, R)$.*

PROOF. By Lemma 9.2, every Red on G's GS-list for A is finished with G in A. Then, by Lemma 9.3, since A strictly dominates A_z, we have $A(G, R) \geq A_z(G, R)$ for any R which is finished with G in A. Suppose R is the last Red subset of G's GS-list, then $A(G, R+) = A_z(G, R+)$. Let R' be the Red subset next to R in G's GS-list in A. Since A dominates A_z, $A(G, R'+) \geq A_z(G, R'+)$. However, $A(G, R) = A(G, R+) - A(G, R'+)$ and $A_z(G, R) = A_z(G, R+) - A_z(G, R'+)$. Therefore, $A(G, R) \leq A_z(G, R)$. Hence, $A(G, R) = A_z(G, R)$. If we take away R from G's GS-list, then R' becomes its last Red subset. By the same argument, we can prove that $A(G, R') = A_z(G, R')$. In this way we can prove all G's partners in A are the same as in A_z.

Definition. A *rotation triple* (G, R, n) is a triple in which (G, R) is a Green-

Red pair of subsets, and n a positive integer. For simplicity, we use (i, j, n) to represent a triple (G_i, R_j, n).

Definition. A *rotation* ϕ is a circular list of rotation triples: (i_0, j_0, n_0), (i_1, j_1, n_1), ..., $(i_{k-1}, j_{k-1}, n_{k-1})$ for some $k \geq 2$ such that, for any $s, t \in [0, k-1]$, $s \neq t$ implies $i_s \neq i_t$ and $j_s \neq j_t$. The first triple (i_0, j_0, n_0) is the successor of the last triple $(i_{k-1}, j_{k-1}, n_{k-1})$.

Definition. Let $\phi = (i_0, j_0, n_0), (i_1, j_1, n_1), ..., (i_{k-1}, j_{k-1}, n_{k-1})$ be a rotation. We define S, the *set of pairs in* ϕ, as (i_0, j_0), (i_1, j_1), ..., (i_{k-1}, j_{k-1}), and T, the *set of shifted pairs in* ϕ, as (i_0, j_1), (i_1, j_2), ..., (i_{k-1}, j_0). Sometimes, we will refer to a pair in the S set of the ϕ as *a pair of* ϕ or *a rotation pair of* ϕ.

Definition. The *rotation matrix* π for a rotation ϕ is a *mxn* matrix, where $\pi_{ij}=1$ if (i, j) is a pair in S; $\pi_{ij}=-1$ if (i, j) is a pair in T; and $\pi_{ij}=0$ for all other cases.

Obviously, any column and any row of the π has either all 0's or exactly one 1, one -1 and the rest 0's. Two distinct rotations may have the same matrices if they differ only in their n's.

Definition. Given a stable matrix A, a rotation triple (G, R, n) and its successor triple (G', R', n'), we define the *criteria* for determining the *next rotation pair* (G', R') as follows: R' is the first Red subset after R available to G, and G' is worst(R', A).

Obviously, G = worst(R, A) for every pair (G, R) of ϕ since ϕ is a circular list of rotation triples.

Definition. Let A be a stable matrix and ϕ a rotation. ϕ is *applicable* to A

iff for any triple (G, R, n) in ϕ, $A(G, R) = n$, $G =$ worst(R, A) and if (G', R', n') is the successor of (G, R, n), then R' is the first Red subset after R available to G. In other words, ϕ is *applicable* to A iff for any triple (G, R, n) in ϕ, $A(G, R) = n$, and if (G', R', n') is the successor of (G, R, n), then (G', R') must follow the criteria specified above for the next rotation pair. A/ϕ stands for the matrix obtained by applying ϕ to A, i.e. A-π.

Suppose L and L' are two ordered sequences of rotations and $L' = \phi, L$. Then, if ϕ is applicable to some A, and $A' = A/\phi$, then $A/L' = A'/L$.

LEMMA 9.3.1. *Let $A'=A/\phi$. Then the GS-lists for A' may be obtained from those of A by updating the GS-lists of A as follows: (1) for any pair (i, j) in the S set of ϕ, if $A'_{ij}=0$ then R_j is removed from G_i's GS-list and G_i is removed from R_j's GS-list; and (2) for any pair (i, j') in the T set of ϕ, if G_i becomes worst$(R_{j'}, A')$ then $R_{j'}$ is marked as finished in G_i's GS-list and if $R_{j'}$ becomes worst(G_i, A') then G_i is marked as finished in $R_{j'}$'s GS-list.*

LEMMA 9.4. *Given a stable matrix A, let ϕ be a rotation which meets the criteria for determining the next pair in the rotation with regard to A. If (G, R) is a rotation pair in ϕ, then R must be finished with G in A.*

PROOF. Since G is worst(R, A), R must be finished with G in A.

Definition. Suppose A and A' are two stable matrices for a given matching instance. A' is an *immediate successor* of A iff A strictly dominates A' from Green's point of view and there is no $A^* \epsilon A$ such that A strictly dominates A^* and A^* strictly dominates A'. The relation of immediate successor between two stable matchings is defined through their matrices.

LEMMA 9.5. *Let A and A' be two stable matrices for a given matching instance such that A strictly dominates A' from Green's point of view. Suppose there is a pair (G, R_0) such that $G = worst(R_0, A)$ and $A(G, R_0) > A'(G, R_0)$. Suppose there is an R_1 which is the first Red subset after R_0 available to G. Let G have one of its slots change its partner from R_0 to R_1. Then each partial sum of G after the change is at least as great as its counterpart in A'.*

PROOF. Since A strictly dominates A' from Green's point of view, by Theorem 3.2, each partial sum of G before the change is at least as great as its counterpart in A'. Precisely, in G's GS-list for A, we have $A(G, R+) \geq A'(G, R+)$ for any R from the top of the list to R_0 (not including R_0); $A(G, R+) > A'(G, R+)$ for any R which is R_0 or strictly between R_0 and R_1, since $A(G, R_0) > A'(G, R_0)$ and, by Lemma 9.3, $A(G, R) \geq A'(G, R)$ if R is finished with G; $A(G, R+) \geq A'(G, R+)$ for rest of the R on the list. Let the new matching after the change be A''. Then $A''(G, R_0) = A(G, R_0) - 1$ while $A''(G, R_1) = A(G, R_1) + 1$. Obviously, $A''(G, R+) = A(G, R+) - 1$ for any R which is R_0 or strictly between R_0 and R_1 and $A''(G, R+) = A(G, R+)$ for all others. Therefore, for any R on G's GS-list for A, we have $A''(G, R+) \geq A'(G, R+)$.

THEOREM 9.1. *Let $\mathcal{A}$ be the solution space for a given stable matching instance, A_z be the Red-Optimal matrix, and $A \in \mathcal{A}$ be any stable matrix other than A_z. Suppose i_1 is any row and j_1 any column in A such that $A_{i1,j1} > A^z_{i1,j1}$. We can build a list in the following way: Let (i_1, j_1) be the first pair in the list. Let (i, j) be any pair* *in the list. The next pair (i', j') after (i, j) is determined by the criteria defined above for determining the next rotation pair. We stop whenever we encounter a new pair (i_0, j_0) which is already in the list. Let S be the sequence of pairs in the list from the pair (i_0, j_0) to the end. Replace each pair (i, j) in S with (i, j, A_{ij}). Then we get ϕ and ϕ is applicable to A. Let T be as in its definition. Then in ϕ every Green slot gets a worse partner in T than in S and every Red slot gets a better partner in T than in S. Furthermore, let $A' = A/\phi$. Then $A' \epsilon \mathcal{A}$.*

PROOF. (1) We prove the ϕ obtained in this way is a rotation. First we prove that, given a pair (i, j) in the list, by following the criteria, the next pair (i', j') always exists and $i' \neq i$ and $j' \neq j$. This is true with (i_1, j_1). By Lemma 7.1, the total number of slots of G_{i1} matched in A is the same as that in A_z. Since A dominates A_z, $A_{i1,j1} > A^z_{i1,j1}$ implies that $A(G_{i1}, R_{j1}+) > A_z(G_{i1}, R_{j1}+)$, which implies there is some R after R_{j1} available to G_{i1} after A is obtained. Let the first such R be R_{j2} and let worst(R_{j2}, A) be G_{i2}. Since G_{i2} is worst(R_{j2}, A), then R_{j2} must be finished with G_{i2}. So we have (i_2, j_2), where $i_2 \neq i_1$ and $j_2 \neq j_1$. Suppose (i', j'), (i, j) are any two consecutive pairs in the list which satisfy criteria. Now we claim there is always some R which is after R_j on G_i's GS-list available to G_i. Suppose there is not. Then either R_j is the last R on G_i's GS-list or all G_i's partners after R_j are finished with G_i. In both cases G_i's GS-list ends with an R finished with G_i. By Corollary 9.2, $A_{i,j} = A^z_{i,j}$. This implies R_j has no replaceable slots for any G since G_i is the worst(R_j, A). This contradicts the assumption that R_j is available to $G_{i'}$. So the claim holds. Since every R is finished with some G, there is always some G which is worst(R, A). So we have (i'', j''), where $i'' \neq i$ and $j'' \neq j$. Then, since the number of Green subsets and the number of Red subsets are finite, some pair in the list must appear again. It is obvious that for every triple (G, R, n), $n = A(G, R)$ is a positive integer since G = worst(R, A). So ϕ is a rotation. It is applicable to A by its construction.

(2) We show that in the rotation ϕ every Green slot gets a worse partner in T than in S and every Red slot gets a better partner in T than in S. This is trivial since every pair in the circular list of ϕ matches the criteria of selecting next pair in ϕ, in which every Green subset moves down its GS-list and every Red subset gives away one slot of its lowest ranked partner and pairs with a higher ranked Green slot.

(3) We prove A' is stable. Suppose it is not. Then there is a blocking pair (g, r) in A'. Let $g \in G$ and $r \in R$.

(a) We prove that g can not be in ϕ. Otherwise, g's A'-partner r' must come from a Red subset, say R', which is the first one after g's A-partner available to G. Then g prefers r to r' implies R is before R' on G's GS-list and R is finished with G. After the ϕ is applied and the A' is obtained, R either remains to be finished with G, if it is not involved in ϕ, or has one of its slots change partners from G to some other Green, if g's A-partner also belongs to R. In the first case r must have a A'-partner at least as good as g since G is worst(R, A'). In the latter case, (g, r) belongs to A and, by (2), r must get its A' partner ranked higher than G. Therefore (g, r) can not be a blocking pair to A' if g is in ϕ.

(b) Since A is stable, it is impossible for both g and r to be outside of ϕ. If

g is not in ϕ and r is in ϕ, then (g, r) is a blocking pair to A too. By (2), r's A'-partner is better than its A-partner. So r prefers g to its A'-partner implies r prefers g to its A-partner. Since g's partners are the same in A and A', g prefers r to its A'-partner implies g prefers r to its A-partner. Since A is stable, this case is also a contradiction.

In summary, A' is stable. That is, $A' \epsilon \mathcal{A}$.

COROLLARY 9.3. *Let* $\phi = (i_0, j_0, n_0), (i_1, j_1, n_1), ..., (i_{k-1}, j_{k-1}, n_{k-1})$ *be a rotation applicable to a stable matrix* A *and* $min(n_0, n_1, ..., n_{k-1}) > 1$. *Then* $\phi' = (i_0, j_0, n_0\text{-}1), (i_1, j_1, n_1\text{-}1), ..., (i_{k-1}, j_{k-1}, n_{k-1}\text{-}1)$ *is a rotation applicable to* A/ϕ.

LEMMA 9.6. *Let* A *and* A' *be two stable matrices and* ϕ *a rotation applicable to* A. *Suppose from Green's point of view,* A *strictly dominates* A' *and* A' *dominates* A/ϕ. *Let* $(G_0, R_0, n_0), (G_1, R, n_1)$ *be two consecutive triples in* ϕ. *If* $A(G_0, R) < A'(G_0, R)$, *then* $A(G_1, R) > A'(G_1, R)$.

PROOF. As in the proof in Lemma 9.3, we have $G' \preceq_R G_1$ where $G' =$ worst(R, A'). If G' is before G_1, then $A'(G_1, R) = 0 < A(G_1, R)$. If G' is G_1, then $A(R, G_1+) = A'(R, G_1+)$ since A and A' are full matchings and G_1 is the worst partner of R in both A and A'. If G_0 is the first Green in R's GS-list for A, then the condition $A(G_0, R) < A'(G_0, R)$ implies $A(R, G_0+) < A'(R, G_0+)$. Otherwise, let the last Green before G_0 be G''. Since A' strictly dominates A from *Red*'s point of view, we have $A(R, G''+) \leq A'(R, G''+)$. In this case, the condition $A(G_0, R) < A'(G_0, R)$ also implies $A(R, G_0+) < A'(R, G_0+)$. Now we prove for any Green subset, G, between G_0 and G_1 on R's GS-list for A', $A(G, R) \geq A'(G, R)$. Assume

G is the first such Green subset after G_0. Then $A(G, R) = A(R, G+) - A(R, G_0+)$ and $A'(G, R) = A'(R, G+) - A'(R, G_0+)$. Since A strictly dominates A' and A' dominates A/ϕ from Green's point of view, we have $A(R, G+) \leq A'(R, G+) \leq A/\phi(R, G+)$ and $A(R, G_0+) \leq A'(R, G_0+) \leq A/\phi(R, G_0+)$. Since $A(R, G_0+) < A'(R, G_0+)$, we have $A'(R, G_0+) = A/\phi(R, G_0+) = A(R, G_0+) + 1$. By the definition of ϕ, we have $A(R, G+) = A/\phi(R, G+) - 1$ and $A(R, G_0+) = A/\phi(R, G_0+) - 1$. Therefore, $A'(G, R) = A'(R, G+) - A'(R, G_0+) = A'(R, G+) - A(R, G_0+) - 1 \leq A/\phi(R, G+) - A(R, G_0+) - 1 = A(R, G+) + 1 - A(R, G_0+) - 1 = A(G, R)$. Following the same argument, we can prove the assertion holds for the next G. Finally, since $A(R, G_1+) = A'(R, G_1+)$ and, for any Green subset, G, between G_0 and G_1 on R's GS-list for A', $A'(G, R) \geq A(G, R)$, then $A(G_1, R) > A'(G_1, R)$.

LEMMA 9.7. *Let A and A' be two stable matrices and ϕ a rotation applicable to A. Suppose from Green's point of view, A strictly dominates A' and A' dominates A/ϕ. Let $(G, R_0, n_0),(G_1, R_1, n_1)$ be two consecutive triples in ϕ. If $A(G, R_0) > A'(G, R_0)$, then $A(G, R_1) < A'(G, R_1)$.*

PROOF. By the definition of ϕ, R_1 is the first Red subset on G's GS-list for A and available to G. Let the last Red subset before R_1 on the list be R'. By Lemma 9.2, all Red subsets before R_1 on the list are finished with G. By Lemma 9.3, we have $A(G, R) \geq A'(G, R)$ for every R before R_1 on G's GS-list for A. Since $A(G, R_0) > A'(G, R_0)$, we have $A(G, R'+) > A'(G, R'+)$. By the definition of ϕ, we have $A(G, R_1+) = A/\phi(G, R_1+)$. Since A dominates A' and A' dominates A/ϕ, we have $A(G, R_1+) \geq A'(G, R_1+) \geq A/\phi(G, R_1+)$. That means, $A'(G, R_1+) = A(G,$

$R_1+)$. Therefore, $A(G, R_1) < A'(G, R_1)$.

LEMMA 9.8. *Let A and A' be two stable matrices and ϕ a rotation applicable to A. Suppose from Green's point of view, A strictly dominates A' and A' dominates A/ϕ. Suppose (G_0, R_0) is a pair in ϕ such that $A(G_0, R_0) > A'(G_0, R_0)$. Then for any pair (G, R) in the ϕ we have $A(G, R) > A'(G, R)$.*

PROOF. Suppose (G_1, R_1) is the successor pair in ϕ to (G_0, R_0). By Lemma 9.7, $A(G_0, R_1) < A'(G_0, R_1)$. Then, by Lemma 9.6, $A(G_1, R_1) > A'(G_1, R_1)$. In this way we can prove every pair in ϕ has the desired property since all pairs form a circular list.

LEMMA 9.9. *Let A be a stable matrix and ϕ be a rotation applicable to A. Then A/ϕ is an immediate successor of A.*

PROOF. Suppose there is a stable matrix A' such that A' is an immediate successor of A and A' strictly dominates A/ϕ. Since ϕ is applicable to A, then, for any G in ϕ, there are two Red subsets R_1 and R_2 such that R_1 is before R_2 on G's GS-list for A and $A(G, R_1) = A/\phi(G, R_1) + 1$ and $A(G, R_2) = A/\phi(G, R_2) - 1$. So, for any R which is R_1 or strictly between R_1 and R_2, $A(G, R+) = A/\phi(G, R+) + 1$ while $A(G, R+) = A/\phi(G, R+)$ for the rest of the *Red* on G's GS-list for A and for any (G, R) which is not in ϕ. Since A dominates A' and A' dominates A/ϕ, then, by Theorem 3.2, for any G in the ϕ and for any R which is R_1 or strictly between R_1 and R_2 (R_1 and R_2 are the same as defined above), $A'(G, R+)$ is either $A(G, R+)$ or $A/\phi(G, R+)$, while $A'(G, R+) = A(G, R+) = A/\phi(G, R+)$ for the rest of the *Red* on G's GS-list for A and for any (G, R) which is not in ϕ. By Lemma 9.8, if there is a pair (G_0, R_0) in ϕ such that $A(G_0, R_0) > A'(G_0, R_0)$, then for any pair (G, R) in the ϕ we have $A(G, R) > A'(G, R)$. Suppose there is no such pair (G_0, R_0) in ϕ that $A(G_0, R_0) > A'(G_0, R_0)$. That means $A(G, R) \leq A'(G, R)$ for any pair (G, R) in ϕ. Since A dominates A', then $A(G, R) = A'(G, R)$ for any pair (G, R) in ϕ, which contradicts the assumption that A' is an immediate successor of A. So, there is some pair (G_0, R_0) in ϕ such that $A(G_0, R_0) > A'(G_0, R_0)$. Then, since A dominates A', we have $A(G, R+) > A'(G, R+)$ which implies $A'(G, R+)$ is $A/\phi(G, R+)$ for any G in ϕ and for any R which is R_1 or strictly between R_1 and R_2 (R_1 and R_2 are the same as defined above). Therefore, for any G we have $A'(G, R+) = A/\phi(G, R+)$ for any R on G's GS-list for A. That implies $A'=A/\phi$. So, the claim holds.

THEOREM 9.2. *Suppose A and A' are two stable matrices for a given instance, and A' is an immediate successor of A, then there is exactly one rotation ϕ applicable to A such that $A' = A/\phi$.*

PROOF. Since A strictly dominates A', then we have at least a pair (G', R') such that $A(G', R') > A'(G', R') \geq A_z(G', R')$. By Theorem 9.1, given A and the pair (G', R'), we can construct a rotation ϕ such that A/ϕ is stable and A strictly dominates A/ϕ from Green's point of view. Given $A(G', R') > A'(G', R')$, by using Lemma 9.7 and Lemma 9.6 repeatedly, we can prove $A(G, R) > A'(G, R)$ for any pair (G, R) in the list as specified in Theorem 9.1, and therefore, for any pair in ϕ. Then, by Lemma 9.5, A/ϕ also dominates A' from Green's point of view since ϕ, A and A' match the conditions in the lemma. This is a contradiction to the fact that A' is an immediate successor of A if $A/\phi \neq A'$. So $A' = A/\phi$. Let π be the rotation matrix of ϕ. Then $\pi = A - A'$ is uniquely determined, so is the S set of ϕ. Then for every triple (G, R, n) in ϕ, $n = A(G, R)$ is uniquely determined. Therefore, ϕ is unique.

In fact, it can be proved that the D and D' of A and A' are the S and T of a rotation ϕ such that $D' = D/\phi$, where D/ϕ stands for the result of applying ϕ to D. If the matrix A and its immediate successor A' are both known, we next describe an easy way to construct the rotation ϕ from A to A'. First, determine the D and D' of the A and A'. Start ϕ with any pair in D. Suppose (g, r) is the last pair in ϕ. Find (g, r') in D' and append (g', r') in D to the ϕ until the first pair of ϕ reappears. Then replace every pair (i, j) on the list with (i, j, A_{ij}).

COROLLARY 9.4. *Suppose A is a stable matrix and A' is an immediate successor of A. If there is a pair (G, R) such that $A(G, R) > A'(G, R)$, then $A(G, R) = A'(G, R) + 1$ and there is exactly one rotation ϕ applicable to A such that $(G, R) \in \phi$ and $A' = A/\phi$. Likewise, if there is a pair (G, R) such that $A(G, R) < A'(G, R)$, then $A(G, R) = A'(G, R) - 1$ and there is exactly one rotation ϕ applicable to A such that (G, R) is in the T of ϕ and $A' = A/\phi$.*

PROOF. Since the rotation matrix of ϕ, π, is $A - A'$, then (G, R), as a part of the difference, must be in the S or T of ϕ.

Definition. If (G, R, n) and (G', R', n') are two successive triples in a rotation ϕ, then we say, ϕ rotates R *down* to R' (for G) and ϕ rotates G' *up* to G (for R'), to indicate that *one slot* of G or R' has moved between two subsets.

COROLLARY 9.5. *For a stable matching instance, no rotation can be* *contained in any other.*

Definition. A *sequence of rotations from A to A'* (A dominates A') is a sequence of rotations, $(\phi_1, \phi_2, ..., \phi_k)$, such that $A_1 = A_0/\phi_1$, $A_2 = A_1/\phi_2$, ..., $A_k = A_{k-1}/\phi_k$, where $A = A_0$, $A' = A_k$, and A_{i+1} is an immediate successor of A_i for any $i \in [0, k-1]$.

COROLLARY 9.6. *Let* $\mathcal{A}$ *be the solution space for a given stable matching instance, A, A'* $\in \mathcal{A}$ *and A dominate A'. Then there is at least one sequence of rotations from A to A'.*

PROOF. Since A dominates A', there is at least one path from A to A' via a sequence of immediate successors of A and its successors on the path. By Theorem 9.2, each stable matrix and its immediate successor uniquely determines a rotation. Therefore, the desired sequence of rotations exists.

Definition. Suppose A and A' are two stable matrices for a given matching instance and A dominates A'. The *Difference Matrix from A to A'* is $Q = A - A'$.

By Corollary 9.6, there is a sequence of rotations $(\phi_1, \phi_2, ..., \phi_k)$ from A to A'. Let π_i be the matrix of ϕ_i for any $i \in [1, k]$, then $Q = \sum_{i-1}^{k} \pi_i$.

COROLLARY 9.7. *Let ϕ_1 and ϕ_2 be two distinct rotations applicable to a stable matrix A. Then the two rotations have no pair in common (we say they are disjoint). Furthermore, ϕ_2 is applicable to A/ϕ_1, ϕ_1 is applicable to A/ϕ_2, and* $A/\phi_1/\phi_2 = A/\phi_2/\phi_1$.

PROOF. Suppose there is a pair in common to two rotations. By the criteria for determining the next rotation pair and the same GS-list for A, the next rotation pair must be in common too. Thus, the two rotations have the same S. Since both ϕ_1 and ϕ_2 are applicable to A, the corresponding n values for each pair in S must be the same in both. Thus, the two rotations are identical, contradicting the assumption. (In fact, since they follow the same criteria for determining the next rotation pair, it is easy to prove there is no Green or Red subset in common to them.) Let π_1 and π_2 be the rotation matrices for ϕ_1 and ϕ_2, respectively. Since both ϕ_1 and ϕ_2 are applicable to A and there is no pair in common, every element of A-π_1 is at least as great as that of π_2, and every element of A-π_2 is at least as great as that of π_1. Furthermore, since the two rotations are disjoint, application of one would not change the n values of the other. Since there is no subset in common, the criteria are not changed to the other after each of them is applied. So, ϕ_2 is applicable to A/ϕ_1 and ϕ_1 is applicable to A/ϕ_2. Since $A\text{-}\pi_1\text{-}\pi_2 = A\text{-}\pi_2\text{-}\pi_1$, then $A/\phi_1/\phi_2 = A/\phi_2/\phi_1$.

THEOREM 9.3. *Suppose A is the solution space for a given stable matching instance, A, A' are any two stable matrices in A and A dominates A'. If there are more than one sequence of rotations from A to A', then they are permutations of each other. Hence the sequences have the same length and consist of the same set of rotations.*

PROOF. Let L_1 and L_2 be two distinct sequences of rotations, $(\phi_1, \phi_2, ..., \phi_p)$ and $(\phi'_1, \phi'_2, ..., \phi'_q)$, from A to A', where $p \geq 2$ and $q \geq 2$. That is, $A/L_1 = A/\phi_1/\phi_2/.../\phi_p = A' = A/\phi'_1/\phi'_2/.../\phi'_q = A/L_2$. We will use induction on the size of L_1, p, to prove $p = q$ and any rotation in L_1 must be in L_2 and vice versa. We start with $p=2$, where $A' = A/\phi_1/\phi_2$. If $\phi_1 = \phi'_1$, then $A/\phi_1 = A/\phi'_1$ and A' is an immediate successor of A/ϕ'_1 since A' is an immediate successor of A/ϕ_1. That means there is exactly one rotation ϕ such that $A'=A/\phi'_1/\phi$. In other words, $\phi_2=\phi'_2$ and $L_1=L_2$, contradicting the assumption that $L_1 \neq L_2$. So, $\phi_1 \neq \phi'_1$. By Corollary 9.7, ϕ_1 and ϕ'_1 have no pair in common. However, A' is an immediate successor of A/ϕ_1 and a successor of A/ϕ'_1. Therefore, ϕ_2 must be equal to ϕ'_1. That is, $A'=A/\phi_1/\phi'_1$. Then, by Corollary 9.7, $A/\phi_1/\phi'_1=A/\phi'_1/\phi_1$, which implies A' is an immediate successor of A/ϕ'_1 and $\phi'_2 = \phi_1$. This proves that the assertion is true for $p=2$. Suppose the assertion is true for $p<k$. We prove it is true for $p=k$. If $\phi_1=\phi'_1$, then $A/\phi_1=A/\phi'_1$. By the induction hypothesis, the assertion is true for A/ϕ_1, A', $(\phi_2, ..., \phi_p)$ and $(\phi'_2, ..., \phi'_q)$. Since $\phi_1=\phi'_1$, it is also true for A, A', $(\phi_1, \phi_2, ..., \phi_p)$ and $(\phi'_1, \phi'_2, ..., \phi'_q)$. If $\phi_1 \neq \phi'_1$, then we can find A'' such that $A'' = A/\phi_1/\phi'_1 = A/\phi'_1/\phi_1$. It is obvious that A'' dominates A'. So there exists a sequence of rotations. say L, from A'' to A'. By the induction hypothesis, the assertion is true for A/ϕ_1, A', $(\phi_2, ..., \phi_k)$ and (ϕ'_1, L), and, therefore, true for A, A', $(\phi_1, \phi_2, ..., \phi_k)$ and (ϕ_1, ϕ'_1, L). That means the size of L is $k-2$. Again, by the induction hypothesis, the assertion is true for A/ϕ'_1, A', $(\phi'_2, ..., \phi'_k)$ and (ϕ_1, L), and, therefore, true for A, A', $(\phi'_1, \phi'_2, ..., \phi'_k)$ and (ϕ'_1, ϕ_1, L). Since the assertion is true for (ϕ_1, ϕ'_1, L) and (ϕ'_1, ϕ_1, L), then the assertion is true for A, A', $(\phi_1, \phi_2, ..., \phi_k)$ and $(\phi'_1, \phi'_2, ..., \phi'_k)$. Thus, the assertion holds.

COROLLARY 9.8. *A rotation can not be equivalent to a sequence of two or more rotations for the same matching instance. That is, for any rotation ϕ applicable to some A, there is no sequence of rotations, $\phi_0, \phi_1, ..., \phi_i$, such that $i \geq 1$, ϕ_0 is* *applicable to A and* $A/\phi = A/\phi_0/\phi_1/.../\phi_i$.

Definition. The *set of rotations of A* is the set of rotations which appear in any sequence from A_0 to A.

Definition. Let A_0 and A_z be the Green-Optimal and the Red-Optimal matrix for a given stable matching instance, respectively. Any sequence of rotations from A_0 to A_z is called a *maximal chain.* A *rotation for the matching instance* is any rotation which occurs in any maximal chain.

THEOREM 9.4. *Any maximal chain contains all the possible rotations for the matching instance.*

PROOF. For any element A in $(A, \leq)$, we have $A_0 \leq A$ and $A \leq A_z$. Therefore, any element of the distributive lattice is on a path from A_0 to A_z. That means every rotation is on a maximal chain. By Theorem 9.3, all maximal chains have the same set of rotations. Thus the theorem is proved.

LEMMA 9.10. *Let ϕ be a rotation containing (G, R, n) and applicable to a stable matrix A. Let A' be a successor of A/ϕ. Then* $n = A(G, R) > A'(G, R)$.

PROOF. By definition, after the application of ϕ to A, we have $A/\phi(G, R) = A(G, R) - 1$ and worst$(R, A/\phi) \preceq_R G$. Precisely, when $n = 1$, then worst$(R, A/\phi) \prec_R G$ and $A/\phi(G, R) = 0$; otherwise, G = worst$(R, A/\phi)$ and $A/\phi(G, R) = n - 1$. By Lemma 9.3, we have $A/\phi(G, R) \geq A'(G, R)$. So, in all cases, $n = A(G, R) > A'(G, R)$.

The above lemma says, for any pair (G, R) in ϕ, the number of matching slots between G and R can not increase once it starts to decrease.

By the way, if A dominates A' from Green's point of view, and (G, R) is on the GS-lists for A, then, if R is finished with G in A, R may remain finished with G in A' or be removed from the GS-lists for A' because of the updates by some rotation; or, if R is not finished with G in A, it may remain unfinished with G in G's GS-list for A' or change to be either finished or removed from the lists.

THEOREM 9.5. *Every rotation for a given matching instance can occur only once on any maximal chain.*

PROOF. By Lemma 9.10, after applying a rotation ϕ containing (G, R, n) to a stable matrix A, the number of matching slots between G and R can not increase but must decrease. This is obvious since this number can grow only when R is not finished with G, and it can decrease only when R is finished with G. So, for any triple (G, R, n) in ϕ, even if $n > 1$, ϕ is not applicable to A/ϕ and its successors since this number is not equal to n any more.

Note that, in the standard stable matching problem, n is always one for any pair in a rotation and, therefore, can be dropped from the definition of rotation and the algorithm.

THEOREM 9.6. *For a given matching instance, let the set Green be partitioned into m non-overlapping subsets, and the set Red into n non-overlapping subsets and let $p = min(m, n)$. Let the size of Green be k and the size of Red be l and let $q = min(k, l)$. Then the maximum number of rotations this instance can have in its solution space is $(p-1)q/2$.*

PROOF. If (g, r) is a pair in the S set of a rotation ϕ, ϕ moves g at least one position down on its preference list, and moves r at least one position up in its preference list. Suppose $g \epsilon G_i$ and G_i has s_i slots. Since q is the maximum number of pairs of slots we can have for a matching, then there are at most $q\text{-}s_i$ positions available on G_i's preference list for all possible rotations. Let X be the total positions available for all possible rotations, then X is not greater than the sum of $q\text{-}s_i$ over i from 1 to m. Since the sum of s_i over i from 1 to m is q, then X is not greater than $qm\text{-}q$. By a symmetrical argument on Red, X is not greater than $qn\text{-}q$. Since $p = min(m, n)$, then X is not greater than $(p\text{-}1)q$. Since each rotation has at least two pairs in its S set, the total number of rotations this instance can have in its solution space is not greater than $(p\text{-}1)q/2$.

Note that, if we let $p=q=n$, then this number is $(n\text{-}1)n/2$, the number for the standard case. Therefore, this formula is a generalization of the standard case.

The Correctness and Time Complexity of Algorithm II:

Algorithm II was originally designed by Gusfield and Irving for the standard matching problem. We modified it to fit our generalized situation. This algorithm uses the results from Algorithm I, namely A_0 , A_z , and the initial GS-lists, to generate all rotations for the given matching instance in the order of their appearance in one of the maximal chains. Its design is based on the principles established in Theorem 9.1, Lemma 9.9, Theorem 9.4, and Theorem 9.5. The algorithm starts by comparing A_0 and A_z to find a pair (i, j) such that $A_{ij}^0 > A_{ij}^z$, and puts the triple (i, j, A_{ij}) onto a stack. By Theorem 9.1, given any stable matching A other than A_z, if there is an element (i, j) in A such that $A_{ij} > A_{ij}^z$, then we can find a rotation ϕ which applies to A. After a ϕ has been found, the triple (i, j, t) is either in ϕ or remains in the stack. If it remains in the stack, then $next((i, j, t))$ in A/ϕ is the same as that in A since the GS-list of G_i for A/ϕ is the same as that for A (Recall G's GS-list is updated only if G is in ϕ). If (i, j, t) is on the top of the stack, then the search for next rotation will continue with A/ϕ. If the triple (i, j, t) is in ϕ, then $A/\phi_{ij} = A_{ij} - 1$ and the search for next triple (i', j', t') will resume at position (i, j) but in A/ϕ rather than A. By Corollary 9.9, A/ϕ is an immediate successor of A. This guarantees that the chain we are constructing is maximal. This search must terminate since A dominates A/ϕ and there is only a finite number of stable matchings between A and A_z. By Theorem 9.4 and Theorem 9.5, since every rotation for the given matching instance occurs exactly once on a maximal chain, we are guaranteed to find all the rotations without repetition. Therefore, the algorithm is correct.

For the time analysis, we assume the time for $next((i, j, t))$ is a constant since GS-lists are used. In addition, for the convenience of discussion, we divide the time complexity of the whole algorithm as the sum of the following parts:

(1) the total number of comparisons needed for searching all pairs, (i, j), such that $A_{ij} > A_{ij}^z$, which is mq in the worst case (the notation follows that in Theorem 9.6).

(2) the total number of times for pushing and popping rotation triples, which

is a multiple of the total number of triples in the rotations. Since each triple corresponds to a rotation, this number is a multiple of the total number of rotations. Suppose the maximum size of a rotation in this instance is c. By Theorem 9.6, the maximum number of rotations an instance can have is $(p-1)q/2$. So, the total number of stack operations is $c(p-1)q$.

(3) the number of assignments to build all rotation matrices, which is the same as that in (2) in the worst case.

(4) the total number of comparisons and assignments needed to update the GS-lists, which is also a multiple of the total number of triples in the rotations.

In summary, the time complexity is $O(mq)$ which is $O(q^2)$ for the standard case.

Algorithm II :

```
{ find A_0 and A_z , and create the initial GS-lists by using Algorithm I.}
{ if A_0 ≠ A_z then do the following.}

set up an empty stack;
k := 0;        i := 1;        j := 1;
while i ≤ m do  { m and n are the number of subsets in Green and Red, respectively}

begin
    if the stack is empty then
    begin
        find := false;
        while (i ≤ m) and (find ≠ true) do
        begin
            while (j ≤ n) and (A_ij^k ≤ A_ij^z) do j := j + 1;
            if j ≤ n then     {A_ij^k > A_ij^z and A_ij^k≠0}
            begin
                push (i, j, A_ij^k) onto the stack;
                find := true
            end
            else
            begin
                i := i + 1;
                j := 1;
            end
        end
    end;  { (i, j) is the first Green-Red pair in the search sequence with A_ij^k ≠ 0
          and A_ij^k > A_ij^z }

    if the stack is not empty then
    begin
        (i, j, t) := the Green-Red-integer triple on top of the stack;
        (i, j, t) := next((i, j, t));
        while (i, j, t) is not in the stack do
        begin
            push (i, j, t) onto the stack;
            (i, j, t) := next((i, j, t))
        end;
        (i', j', t') := top of stack;
        pop stack;
        set up a list S containing only the triple (i', j', t');
        while (i, j, t) ≠ (i', j', t') do
```

```
            begin
                    (i', j', t') := top of stack;
                    pop stack;
                    add the triple (i', j', t') to the head of S
            end;
            output φ_k;
            Build-Matrix(φ_k);   {build matrix Γ^k with φ_k};
            A^(k+1) := A^k - Γ^k;
            k := k + 1;
            GS_Update;  {update the reduced preference lists}
    end
end.
```

```
function next((i, j, n));

begin

        R_t := first R after R_j available to G_i in A^k;
        G_s := worst(R_t , A^k);
        next((i, j, n)) := (s, t, A_st^k);

end.
```

```
Procedure Build-Matrix(φ_k);

begin

        Γ := 0;
        (i, j) := first pair in φ_k;
        repeat
                Γ_ij := 1;
                (i', j') := the pair succeeding (i, j) in φ_k;
                Γ_ij' := -1;
                (i, j) := (i', j');
        until ((i, j) is the last pair in φ_k);

        Γ_ij := 1;
        (i', j') := the first pair in φ_k;
        Γ_ij' := -1;
end.
```

Procedure **GS_Update**; { after new A is obtained }

begin

```
        (i, j) := first pair in φ_k;
        repeat
                if A_ij=0 then
                        delete R_j from G_i's GS-list;
                        delete G_i from R_j's GS-list;
                (i', j') := the pair succeeding (i, j) in φ_k;
                if G_i becomes worst(R_j', A^k) then
                        mark R_j' as finished in G_i's GS-list;
                if R_j' becomes worst(G_i, A^k) then
                        mark G_i as finished in R_j''s GS-list;
                (i, j) := (i', j');
        until ((i, j) is the last pair in φ_k);

        if A_ij=0 then
                delete R_j from G_i's GS-list;
                delete G_i from R_j's GS-list;
        (i', j') := the first pair in φ_k;
        if G_i becomes worst(R_j', A^k) then
                mark R_j' as finished in G_i's GS-list;
        if R_j' becomes worst(G_i, A^k) then
                mark G_i as finished in R_j''s GS-list;
```

end.

Chapter 10. The Rotation Poset (Algorithm III)

By Algorithm II, we can get the set of all rotations for a given matching instance. However, some rotations must be applied before some others in all maximal chains of the solution lattice. That means there is a partial order on the members of the rotation set. Gusfield and Irving [8] prove that there is a one-to-one correspondence between a stable matching and a top closed subset of the rotation poset in the standard case. This must also be true in our case since both solution spaces have basically the same representation as proved in the last section. Formally, we have the following definitions and theorems.

Definition. Let ϕ and ϕ' be any two distinct rotations for a matching instance. Then $\phi < \phi'$, and we say ϕ precedes ϕ', iff ϕ appears before ϕ' in every maximal chain of the solution lattice. ϕ *immediately precedes* ϕ' iff $\phi < \phi'$, and there is no other rotation ϕ'' such that $\phi < \phi'' < \phi'$.

Definition. A rotation ϕ *can not be applied after* a rotation τ iff ϕ can not be applied to any matching obtained by applying rotations to a matching A/τ that τ has been applied to.

Uniqueness Principle. *For any property of rotations, if one rotation with that property can not be applied after another with that property, then there is at most one rotation with that property.*

PROOF. If there are two rotations with the given property, they must both occur in some maximal chain. This contradicts the assumption that one can not be applied after the other.

LEMMA 10.1. *Any rotation triple (G, R, n) occurs in at most one rotation.*

PROOF. By Lemma 9.10, after a rotation containing (G, R, n) has been applied, the number of partners between G and R is decreased and never increases to n again. Thus a second such rotation can never be applied. By the Uniqueness Principle, this rotation is unique in a maximal chain. So, the claim holds.

LEMMA 10.2. *Let ϕ and ϕ' be two rotations for a given matching instance. Suppose ϕ and ϕ' have triple (G, R, n) and (G, R, n'), respectively, and $n > n'$. Then ϕ precedes ϕ'.*

PROOF. By Lemma 9.10, in any maximal chain, once the rotation containing (G, R, n') has been applied, the number of partners between G and R will always be $n' - 1$ or less and thus a rotation containing (G, R, n) cannot be applied after a rotation containing (G, R, n').

LEMMA 10.3. *The unique rotation containing $(G, R, n+1)$ applies to a stable matching A with $A(G, R) = n+1$ and produces a matching B with $B(G, R) = n$.*

PROOF. By Corollary 9.4 and Lemma 10.1.

LEMMA 10.4. *At most one rotation rotates an R's worst partner up to G or above. That is, at most one rotation applies to a stable matching A in which $G \prec_R worst(R, A)$ and produces a matching B in which $worst(R, B) \preceq_R G$.*

PROOF. By Lemma 9.10, rotations always move a Red's worst partner up. Hence two such rotations which move R's worst partner up to G or above cannot be applied one after another. Hence again, the Uniqueness Principle applies.

LEMMA 10.5. *At most one rotation produces a given (G, R, n) with G = R's worst partner. That is, there is only one rotation which applies to a stable matching A with* $A(G, R) \neq n$ *and produces a stable matching B with* $G = worst(R, B)$ *and* $B(G, R) = n$.

PROOF. Suppose ϕ produces a matching B with G = worst(R, B) and $B(G, R) = n$ from a matching A with $A(G, R) \neq n$. Then either $A(G, R) < n$ and ϕ rotates a partner of R up to G = worst(R, B) or $A(G, R) = n+1$. In the first case, there is at most one such rotation by Lemma 10.4, while in the second case, there is at most one such rotation by Lemma 10.3. Finally no two rotations of the above types can apply after each other since after the first, the number of partners between G and R is always n and the second can not apply; and after the second, R's worst partner is always G or higher and the first can not apply.

LEMMA 10.6. *If a rotation* ϕ *applies to a matching A, then it also applies to matchings obtained by applying any sequence of rotations other than* ϕ *to A.*

PROOF. Suppose B is obtained from A by applying some sequence of rotations. If (G, R, n) is in ϕ and ϕ applies to A, then $A(G, R) = n$. Since no rotation other than ϕ contains (G, R, n) (by Lemma 10.1), $B(G, R) = n$. Also G = worst(R, A) and since a rotation never moves a worst partner down and since G is a partner of R in B, G = worst(R, B). For two successive triples (G, R, n) and (G', R', n') in ϕ and any S between R and R' on G's GS-list for A we have $G =_S$ worst(S, A). Since rotations never move a worst partner down, we have worst(S, B) $\preceq_S G$ which means S is either finished with G on G's GS-list for B or has been dropped from it. So ϕ is also applicable to B.

LEMMA 10.7. *If all rotations preceding ϕ have been applied to the Green-Optimal matching (A_0), then ϕ can be applied.*

PROOF. If ϕ can not be applied to a matching A, then for some triple (G, R, n) of ϕ, $G \neq \text{worst}(R, A)$ or $A(G, R) \neq n$ or for some successive triples (G, R, n), (G', R', n') and some S between R and R' on the GS-list of G for A, $G \prec_S \text{worst}(S, A)$. By Lemmas 10.4 and 10.5, there are unique rotations which remedy these obstructions. These rotations must always precede ϕ. Hence if all rotations which precede ϕ have been applied, so can ϕ be applied.

COROLLARY 10.1. *Any sequence of rotations which contains all rotations and preserves the precedence relation, i.e., a topological sorting of the rotations under the precedence relation, is a maximal chain of rotations.*

Definition. A set of rotations is *top closed* with regard to the partial order $<$ iff, for any member of the set, its predecessor is also in the set.

LEMMA 10.8. *A set of rotations is top closed iff it is the top segment of a maximal chain.*

PROOF. $= >$: Suppose W is a rotation poset for a given matching instance and V is a top closed subset of W. Let L_v be a topologically ordered sequence of all rotations in V with regard to the partial order $<$. Then there exists a topologically ordered sequence, L_w, of all rotations in W with L_v as its top segment. By Corollary 10.1, L_w corresponds to a unique maximal chain of the solution space. Therefore, L_v corresponds to the top segment of a maximal chain.

$<=$: By the definition of $<$, the top segment of a maximal chain includes all the predecessors of any element in the segment. Therefore, the set of rotations in the top segment of a maximal chain is top closed.

THEOREM 10.1. *A set of rotations is top closed iff it is the set of rotations for some stable matching A.*

PROOF. $=>$: Suppose V is a top closed set of rotations. By Lemma 10.8, V is the top segment of a maximal chain. By definition, a set of rotations for a stable matching A is the set of rotations which appear in any sequence from A_0 to A. Therefore, V is the set of rotations for some stable matching A.

$<=$: Suppose W is the set of rotations for some stable matching A. Thus W is the set of rotations above A in some maximal chain. By Lemma 10.8, W is top closed.

Now we take a closer look at the immediate predecessors.

Definition. A rotation τ is a *reducing predecessor* of ϕ iff for some triple (G, R, n) of ϕ, $(G, R, n+1)$ is in τ. A rotation μ is an *improving predecessor* of ϕ iff for some successive triples (G, R, n) and (G', R', n') of ϕ and (H', S', m') and (H, S, m) of μ, $R \preceq_G S \prec_G R'$ and $H' \preceq_S G \prec_S H$ and $m = 1$.

Here, a reducing predecessor τ reduces $n+1$ to n and an improving predecessor μ moves S's worst partner up to H' which is G or above so ϕ can move R down to R' over S.

LEMMA 10.9. *If τ is a reducing or improving predecessor of ϕ,* then τ precedes ϕ.

PROOF. This follows from Lemmas 10.3 and 10.4.

LEMMA 10.10. *If all reducing predecessors and improving predecessors of ϕ have been applied to the Green-Optimal matching (A_0), then ϕ can be applied.*

PROOF. Use the same proof in Lemma 10.7 noting that the required predecessors are either reducing or improving predecessors.

THEOREM 10.2. *The predecessor relation on rotations is the transitive closure of the union of the reducing and improving predecessor relations.*

PROOF. Let ϕ be any rotation, let P be the set of predecessors of ϕ under the precedence relation, and let RI^* be the set of predecessors of ϕ under the transitive closure of the union of the reducing and improving relations. By Lemma 10.9, $RI^* \subseteq P$. To prove that $P \subseteq RI^*$, it suffices to exhibit a maximal chain in which the rotations before ϕ are exactly the rotations of RI^*.

Let ϕ_1 , ϕ_2 , ϕ_3 , ... ,ϕ be a topological sorting of RI^* along with ϕ. By Lemma 10.10. these rotations form a chain of rotations from the Green-optimal matching to some matching A. The desired maximal chain is formed by extending this chain below A to a maximal chain.

Now we need to find out, given all rotations of a matching instance, how to find the union of the reducing and improving predecessor relations. By definition, we can find this partial order by enumerating all maximal chains of the solution lattice. However, it can be very inefficient if the problem size is very big. We will find it via Algorithm III without using the enumeration method.

Let A be a stable matrix in a maximal chain and ϕ a rotation. By definition,

ϕ is applicable to A iff for any triple (i, j, n) in ϕ, $A_{ij} = n$, G_i = worst(R_j, A), and if (i', j', n') is its successor triple, then $R_{j'}$ is the first Red subset after R_j available to G_i on G_i's GS-list for A. Suppose ϕ is not applicable to A and ϕ has never been applied to any matrix before A in the chain. We want to check what kind of rotations must precede ϕ in the maximal chain before ϕ can be successfully applied to one of A's successors. The problems with the applicability, and, hence, the predecessor required by the problem, must be one or more of the following:

(1) If $A_{ij} > n$ for any triple (i, j, n) in ϕ, then ϕ can not be applied to A. That means, any rotation ϕ' that reduces A_{ij} to n must precede ϕ. Obviously, ϕ' is a reducing predecessor of ϕ.

(2) If $A_{ij} < n$ for any triple (i, j, n) in ϕ, then ϕ can not be applied to A. That means, any rotation that increases A_{ij} to n by generating a new pair (i, j) must precede ϕ. That is, ϕ and ϕ' have some common pair in the S set of ϕ and the T set of ϕ'. Formally, the relation can be described as: ϕ' precedes ϕ if for some successive triples (G, R, n) and (G', R', n') of ϕ and (H', S', m') and (H, S, m) of ϕ', $R =_G S \prec_G R'$ and $H' =_S G \prec_S H$ and $m = 1$. Obviously ϕ' is an improving predecessor.

In steps (3) - (5), we check how other rotations can help ϕ meet the criteria for determining next rotation pair. Let (i, j, n), (i', j', n') be two consecutive triples in ϕ.

(3) If $R_{j'}$ is not the *first* Red subset after R_j available to G_i, then there must be at least one R between R_j and $R_{j'}$ in G_i's GS-list for A and available to G_i. R can

eventually become finished with G_i when other rotations add more (G_i, R) pairs and make G_i the worst partner of R, or be removed from the list when some rotations move the worst partner of R from below G_i to above G_i in the GS-list of R. Therefore, any other rotation which moves R from below G_i to G_i or above must precede ϕ to avoid instability. Formally, we have: ϕ' precedes ϕ if for some successive triples (G, R, n) and (G', R', n') of ϕ and (H', S', m') and (H, S, m) of ϕ', $R \nprec_G S \nprec_G R'$ and $H' \preceq_S G \prec_S H$ and $m = 1$. Obviously ϕ' is an improving predecessor.

(4) If $G_i \neq \text{worst}(R_j, A)$, then there must be some G_0 after G_i on R_j's GS-list for A, and any rotation that removes the G_0 from the list must precede ϕ. Suppose ϕ' is the last rotation to make G_i the worst partner of R_j. Then ϕ' can remove (G_0, R_j) by moving R_j to G_i or above. If ϕ' moves R_j to G_i, the relation of ϕ' and ϕ is the same as that in (2). Therefore, we only need to find the predecessor that moves R_j above G_i. Formally, the relation can be described as: ϕ' precedes ϕ if for some successive triples (G, R, n) and (G', R', n') of ϕ and (H', S', m') and (H, S, m) of ϕ', $R =_G S \prec_G R'$ and $H' \prec_S G \prec_S H$ and $m = 1$. Obviously ϕ' is an improving predecessor.

(5) If $R_{j'}$ is *not available* to G_i, that means $R_{j'}$ is finished with G_i. Then, there is no rotation that can help since nothing can move $R_{j'}$ from the finished status back to unfinished status with G_i.

The Correctness and Time Complexity of Algorithm III:

Algorithm III is a direct implementation of the definitions of the reducing and improving predecessors and is totally different from Gusfield's, since, as we have discovered before, there are many new situations that only exist in our grouped matching problem. This algorithm uses the output of Algorithms I and II, namely A_0, A_z, the initial GS-lists, and a maximal chain *MC* which lists all the rotations of the given matching instance. By the definition of precedence, if rotation ϕ precedes τ, ϕ occurs before τ in *MC*. Let *I(G,R)* and *II(G,R)* be arrays indexed on Green and Red subsets, *G* and *R*. For each *G* and each *R*, *I(G,R)* will point to a list of all rotations ϕ which contain *(G, R, n)* for some *n*. After *I* is filled, each list will be sorted with respect to the precedence order. Thus they are also sorted in decreasing order with respect to *n*. After *II* is filled, *II(G,R)* is either empty or a single rotation, the unique rotation which contains a triple *(G', R, 1)* and which rotates *G'* to *G* or above. *RIP* is a list of all pairs of rotations $<\tau, \phi>$ such that τ is a reducing or an improving predecessor of ϕ. It is clear from the definition of the reducing and improving precedence relations that all pairs of these relations have been generated and placed in *RIP*.

Let *#T* be the total number of (*G, R, n*) triples which occur in any rotation for a given matching instance. To fill array *I* takes *#T* steps. To fill array *II* also takes *#T* steps, provided one locates *G"* in *R*'s precedence list in constant time. It is easy to implement this using the GS-lists: Place a marker (implemented as a pointer) on each *R*'s initial GS-list, initially at the bottom of the list. When a rotation ϕ of *MC* is encountered which moves *R* up from *G'* to *G''*, the marker will be at *G'*. As the *II(G,R)*'s are appended, move the marker up until it is at *G''* where it will be ready for the next rotation. To Generate *RIP*, the first block takes *#T* steps since the sum of the lengths of the lists of array *I* is *#T*. The second block takes *#T* steps using the method for filling array *II*. In all, the time complexity of the algorithm is *O(#T)*. By Theorem 9.6, *#T* is less than or equal to *(p-1)q/2*. Therefore, the time complexity of the algorithm is *O(pq)*.

Note that, in the standard case, array *I* is not needed, and the conditions specified in array *II* for the improving predecessors can be simpler since several situations do not exist in the standard case.

<u>**Algorithm III**</u> :

{ find A_0 and A_z , and create the initial GS-lists by using Algorithm I}
{ If $A_0 \neq A_z$ then find all rotations $\phi_0, \phi_1, ..., \phi_k$ $(k>0)$ by using Algorithm II}

set up two empty *mxn* arrays, *I* and *II*.

{*I(G, R)* points to a list of all rotations ϕ which contain (*G, R, n*) for some *n*. Each list is sorted with respect to the precedence order. Thus they are also sorted in decreasing order with respect to the *n*.}
{*II(G, R)* is empty or a single rotation, the unique rotation which contains a triple *(G', R, 1)* and which rotates *G'* to *G* or above.}

set up an empty list, *RIP*.
{*RIP* is a list of all rotations $<\tau, \phi>$ such that τ is a reducing or an improving predecessor of ϕ.}

{Fill array *I*}
for $i := 0$ to k do { for each rotation }
  for each $(G, R, n) \epsilon \phi_i$
    append ϕ_i to the list $I(G, R)$.

{Fill array *II*}
for $i := 0$ to k do { for each rotation }
  for each successive pair (G'', R'', n''), $(G', R, 1) \epsilon \phi_i$,
      {hence ϕ rotates *R* from *G'* up to *G''*}
    for each *G* such that $G'' \leq_R G <_R G'$,
      append ϕ_i to $II(G,R)$.

{Generate *RIP*}
for each *G*, each *R*,
for each successive pair $<\phi_i , \phi_j>$ of rotations in the list $I(G,R)$
  append $<\phi_i , \phi_j>$ to *RIP*.

for $i := 0$ to k do { for each rotation }
  for each successive pair (G, R, n), $(G', R', n') \epsilon \phi_i$,
  for each R'' such that $R \leq_G R'' <_G R'$ and $II(G,R'')$ is nonempty,
    append $<II(G,R''), \phi>$ to *RIP*.

Chapter 11. Optimization

The Given:

By Algorithm III, we have found the rotation poset which is represented by a digraph of the rotations for the given matching instance. Now we want to search from A_0 and find the set of rotations for an optimal stable matching with regard to the given objective matrix B. To do so, we need to know the weight of each rotation with regard to B. The formal definition of the weight of a rotation is as follows:

Definition. Given an objective matrix, B, and a rotation $\phi = ((i_0, j_0, n_0), (i_1, j_1, n_1), ..., (i_{k-1}, j_{k-1}, n_{k-1}))$, the *weight* of ϕ with regard to the B, $w(\phi)$, is

$$\sum_{t=0}^{k-1} (b_{i_t j_{t+1}} - b_{i_t j_t}) ,$$

where j_k is j_0 .

So, as inputs to the optimization problem, we have the value of the objective function for A_0 , and a digraph of rotations with their weights.

The Objectives:

Given the inputs as specified above, the optimization problem is to find a maximum-weight top-closed subset of the weighted rotation poset for the given matching instance.

The Method:

Gusfield and Irving [8] recommend the following way to reduce the problem of finding a maximum-weight top-closed subset of the weighted rotation poset to the problem of finding a maximum flow and minimum s-t cut. In this way, we can find a minimum s-t cut K in the digraph in $O(n^4)$ time and use K to find the maximum weight closed subset of the rotation poset in $O(n^2)$ time (n is the size of the problem in the standard case). To be complete we quote the method:

> "A node ϕ in $G(M)$ (*the digraph of the rotation poset*) is called a negative node if $w(\phi) < 0$, and is called a positive node if $w(\phi) > 0$. We let P denote the set of positive nodes and let N denote the set of negative nodes of $G(M)$. For a set S of rotations, we define the weight of S, denoted $w(S)$ to be the sum of the weights of the rotations in S.
>
> Given a weighted, directed network G with designated nodes s and t, an s-t cut is a partition of the nodes of G into two sets (U, V), with $s \in U$ and $t \in V$. The capacity of the cut, denoted $C(U, V)$, is the sum of the weights on the edges which cross from nodes of U to nodes of V. A minimum s-t cut is an s-t cut of minimum total weight.
>
> Given $G(M)$, we define the following capacitated s-t flow network $H(M)$. A source node s and a terminal node t are added to digraph $G(M)$. A directed edge is added from s to every negative node ϕ_i in $G(M)$; the capacity of directed edge (s, p_i) is set to $|w(\phi_i)|$. A directed edge is added to node t from every positive node ϕ_j; the capacity of edge (ϕ_j, t) is $w(\phi_j)$. Note that $H(M)$ contains neither edge (s, ϕ) nor (ϕ, t) if $w(\phi) = 0$. The capacity of every original edge in $G(M)$ is set to be infinite."
>
> Then, Gusfield and Irving proved the following theorems:
>
> "Let K be the set of edges crossing a minimum s-t cut in $H(M)$, and let P_k be the positive nodes of $H(M)$ whose edges into t are not in K. Then the nodes P_k and all nodes that reach them in $H(M)$ (their predecessors), define a maximum-weight closed subset of $G(M)$. Further, P_k is exactly the set of positive nodes of this closed subset of $G(M)$."
>
> "A maximum flow and minimum s-t cut in $H(M)$ can be found in $O(n^4)$ time."
>
> "Given the positive nodes S^* in a maximum-weight closed subset of $G(M)$, the entire closed subset can be found in $O(n^2)$ time."

The Types of Optimization:

The linear objective function of A_{ij} , which is the model for the optimization problem under the stability constraints, can cover the following problems:

(a) Egalitarian stable matching:

Definition. Let *rank(X, Y)* be the position of X in Y's list. An egalitarian table matching is a stable matching which minimizes

$$\sum_{\forall G\ \forall R} A(G,R)(rank(G,R)+rank(R,G))$$

over all stable matchings.

Obviously, a Green-Optimal matching minimizes $\sum_{\forall G\ \forall R} A(G,R)\ rank(G,R)$ and maximizes $\sum_{\forall G\ \forall R} A(G,R)\ rank(R,G)$ over all A in $\boldsymbol{A}$, and the Red-Optimal matching does the opposite.

Let $b_{ij} = (n - rank(G_i, R_j)) + (m - rank(R_j, G_i))$. Then

$$\begin{aligned}
&\max_{\forall A\epsilon \boldsymbol{A}} \sum_{\forall i \forall j} b_{ij} A_{ij} \\
&= \max_{\forall A\epsilon \boldsymbol{A}} \sum_{\forall i \forall j} ((m+n)-rank(G_i,R_j)-rank(R_j,G_i))\ A_{ij} \\
&= (m+n)\sum_{\forall i \forall j} A_{ij} + \max_{\forall A\epsilon \boldsymbol{A}} -\sum_{\forall i \forall j} (rank(G_i,R_j)+rank(R_j,G_i))\ A_{ij} \\
&= (m+n)\sum_{\forall i \forall j} A_{ij} - \min_{\forall A\epsilon \boldsymbol{A}} \sum_{\forall i \forall j} (rank(G_i,R_j)+rank(R_j,G_i))\ A_{ij}.
\end{aligned}$$

So, if A is the stable matching that maximizes $\sum_{\forall i \forall j} b_{ij} A_{ij}$, then A is an egalitarian stable matching, since $\sum_{\forall i \forall j} A_{ij}$ is a constant for all $A \epsilon \boldsymbol{A}$.

(b) Optimal stable matching:

Definition. Given a matching instance, let the preference lists be changed in this way: for any two subsets X and Y, in which one is from the Green and the other from the Red, let *c(X, Y)* be a real number on X's preference list indicating X's preference for Y, such that $c(X, Y) \neq c(X, Y')$ if $Y \neq Y'$, and X prefers Y to Y' iff $c(X, Y) < c(X, Y')$.

Note the only change in an optimal stable matching from an egalitarian one is the preference lists. The definition of stability is not changed.

Definition. An optimal stable matching is a stable matching which minimizes $\sum_{\forall G\ \forall R} A(G,R)(c(G,R)+c(R,G))$ over all stable matchings.

Obviously the egalitarian stable matching problem is a special case of the optimal stable matching, where $c(X, Y) = rank(X, Y)$. Let C be the maximum of $c(X, Y)$ for all X and Y. Let $b_{ij} = 2C - c(G_i, R_j) - c(R_j, G_i)$. Then, similar to the egalitarian stable matching problem, the optimal stable matching problem can be reduced to the stable matching problem in our case.

(c) Resource allocation with bilateral preferences:

Resource allocation problem used to be considered unilaterally from the resource users' point of view. That is, what is the optimal allocation of resources is determined solely by the interests of the activities. However, in many applications, resources can have preferences for activities assigned to them. This preference can either be the subjective initiatives of the resource and its management, or the restrictions on the activities imposed by the objective reality. For example, the management of an information network with multi-level computers would not prefer to let its high resolution graphic terminals be dedicated to word-processing or data entry. On the other hand, management's preference may represent some restrictions. For example, the farmer may prefer to grow some crops to others because of biological limitations of the land and the management cost or practice required by the crops. More generally, we can consider this model as an optimization of an objective function based on the matching of two ranking systems.

In our model, either *Green* or *Red* can be the resources or the activities. Each subset represents a type of resource or a type of activity. The capacity of the resources and the upper limits of the activities can be represented by the number of slots each subset has. Without loss of generality, let *Green* be the activities. Let $e_{ij} \geq 0$ represent the coefficient of efficiency for the activity-resource pair G_i and R_j. Let $p_i \geq 0$ be the return value of activity G_i. Then $B_{ij} = p_i e_{ij}$ and the problem can be solved.

Chapter 12. Conclusions

To meet the requirements of a special kind of resource allocation problem, involved with matching diverse, multiple resources with activities based on the bilateral preferences and optimizing an objective function over the matchings, this study has defined a generalized model for the stable marriage problem, provided its supporting theory, modified some existing algorithms and created a new algorithm to make the solutions computationally feasible. Given such a resource allocation problem with grouped preference lists, we first apply Algorithm I to get maximal and minimal elements of the solution lattice, A_0 and A_z. Then we use Algorithm II to find all rotations of the solution space. Then Algorithm III was used to find the partial order relations in the set of rotations. Finally, we reduced this optimization problem to a minimum cut problem. The whole process runs in time polynomial in the problem size and the solutions are guaranteed to have Pareto Efficiency and the majority assignment property. We found that two optimization problems in the stable marriage problem are also covered under this model.

Appendix. An Example

Given:

Preference Lists: (Each preference list is in the descending order of preferences. The number in () is the size of the subset.)

G1 (4) :	R3, R2, R1, R4, R5.	R1 (2) :	G1, G2, G3, G4.
G2 (3) :	R2, R1, R3, R5, R4.	R2 (2) :	G1, G2, G4, G3.
G3 (1) :	R5, R4, R1, R2, R3.	R3 (3) :	G2, G1, G4, G3.
G4 (1) :	R4, R5, R2, R1, R3.	R4 (2) :	G3, G4, G1, G2.
		R5 (2) :	G4, G3, G2, G1.

Objective Matrix, B:

	R1	R2	R3	R4	R5
G1	1.5	.9	.85	.2	.4
G2	.92	.85	.7	.4	.3
G3	.1	.05	.2	.8	1.2
G4	.23	1.1	1.4	1.7	1.3

The Maximal Elements of the Solution Lattice: (obtained via Algorithm I)

M_0:

	R1 (2)	R2 (2)	R3 (3)	R4 (1)	R5 (1)
G1 (4)	0	1	3	0	0
G2 (3)	2	1	0	0	0
G3 (1)	0	0	0	0	1
G4 (1)	0	0	0	1	0

M_7:

R1 (2)	R2 (2)	R3 (3)	R4 (1)	R5 (1)
2	2	0	0	0
2	0	3	0	0
0	0	0	1	0
0	0	0	0	1

All Rotations for the Given Instance: (obtained via Algorithm II)

ϕ_0 : $\{(1, 3, 3), (2, 2, 1)\}$ ϕ_1 : $\{(3, 5, 1), (4, 4, 1)\}$
ϕ_2 : $\{(1, 3, 2), (2, 1, 2)\}$ ϕ_3 : $\{(1, 3, 1), (2, 1, 1)\}$

The Partial Order of the Rotation Poset: (obtained via Algorithm III)

ϕ_0 pred ϕ_2 pred ϕ_3

The Weights of Rotations:

ϕ_0 : -0.1 ϕ_1 : -0.8 ϕ_2 : 0.43 ϕ_3 : 0.43

The Maximum-Weight Top-Closed Subset of Rotations & the Optimal Matching:

$$M_0 \xrightarrow[\phi_0]{} M_1 \xrightarrow[\phi_2]{} M_3 \xrightarrow[\phi_3]{} M_5$$

Solution Space:

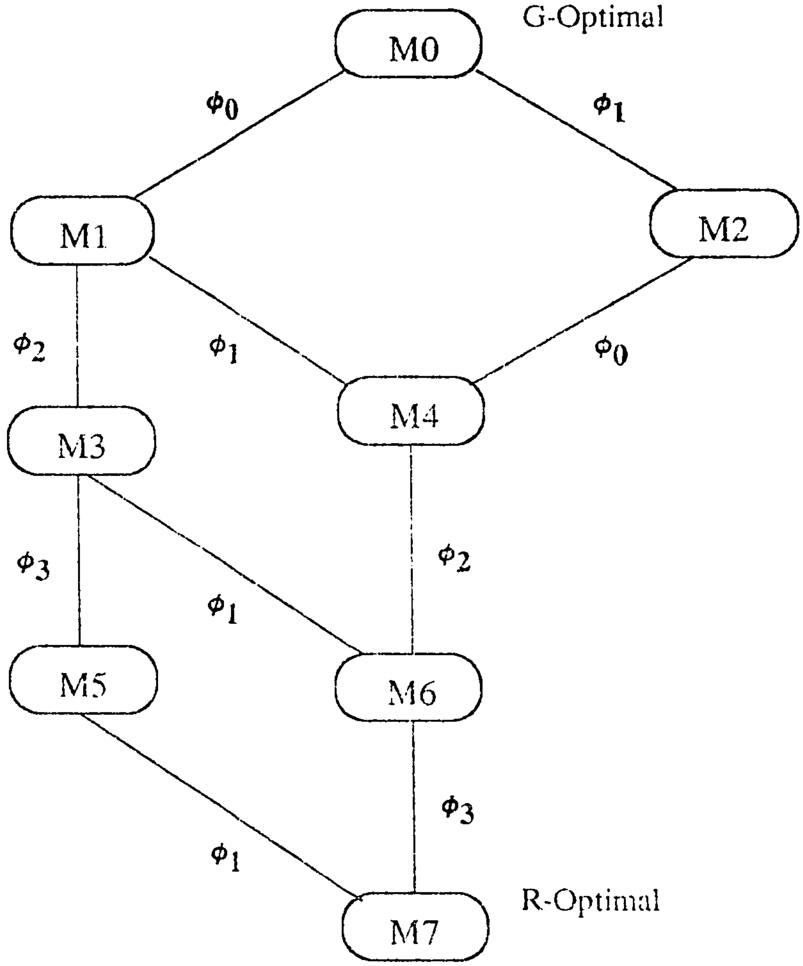
		R1	R2	R3	R4	R5		R1	R2	R3	R4	R5
		(2)	(2)	(3)	(1)	(1)		(2)	(2)	(3)	(1)	(1)
M_0 :	G1 (4)	0	1	3	0	0	M_1 :	0	2	2	0	0
	G2 (3)	2	1	0	0	0		2	0	1	0	0
	G3 (1)	0	0	0	0	1		0	0	0	0	1
	G4 (1)	0	0	0	1	0		0	0	0	1	0
M_2 :	G1 (4)	0	1	3	0	0	M_3 :	1	2	1	0	0
	G2 (3)	2	1	0	0	0		1	0	2	0	0
	G3 (1)	0	0	0	1	0		0	0	0	0	1
	G4 (1)	0	0	0	0	1		0	0	0	1	0
M_4 :	G1 (4)	0	2	2	0	0	M_5 :	2	2	0	0	0
	G2 (3)	2	0	1	0	0		0	0	3	0	0
	G3 (1)	0	0	0	1	0		0	0	0	0	1
	G4 (1)	0	0	0	0	1		0	0	0	1	0
M_6 :	G1 (4)	1	2	1	0	0	M_7 :	2	2	0	0	0
	G2 (3)	1	0	2	0	0		0	0	3	0	0
	G3 (1)	0	0	0	1	0		0	0	0	1	0
	G4 (1)	0	0	0	0	1		0	0	0	0	1

$M_0 / \phi_0 = M_1$; $M_0 / \phi_1 = M_2$; $M_1 / \phi_1 = M_4$; $M_1 / \phi_2 = M_3$;
$M_2 / \phi_0 = M_4$; $M_3 / \phi_1 = M_6$; $M_3 / \phi_3 = M_5$; $M_4 / \phi_2 = M_6$;
$M_5 / \phi_1 = M_7$; $M_6 / \phi_3 = M_7$.

The Values of the Objective Function for Each Stable Matching:

M_0 :	9.04	M_1 :	8.94	M_2 :	8.24	M_3 :	9.37
M_4 :	8.14	M_5 :	9.8	M_6 :	8.57	M_7 :	9

The Distributive Lattice of Stable Matchings



References

1. BLAIR, C. The lattice structure of the set of stable matchings with multiple partners. *Mathematics of Operations Research*, 13 (1988), 619-628.

2. FEDER, T. A new fixed point approach for stable networks and stable marriages. Extended Abstract. *Twenty-First Annual Symposium on Theory of Computing*, (1989).

3. GALE, D. AND SHAPLEY, L. S. College admissions and the stability of marriage. *American Mathematical Monthly*, 69 (1962), 9-15.

4. GÄRDENFORS, P. Match making: assignments based on bilateral preferences. *Behavioral Science*, 20 (1975), 166-173.

5. GOLDBERG, A. AND TARJAN, R. E. A new approach to the maximum flow problem. In *Proceedings of the 18th ACM Symposium on the Theory of Computing*, (1986), 136-146.

6. GRATZER, G. *Lattice Theory*. Freeman, San Francisco, (1971).

7. GUSFIELD, D. Three fast algorithms for four problems in stable marriage. *SIAM Journal on Computing*, 16 (1987), 111-128.

8. GUSFIELD, D. AND IRVING, R. W. *The Stable Marriage Problem: Structure and Algorithms*. The MIT Press, (1989).

9. HWANG, J. S. The algebra of stable marriages. *Intern. J. Computer Math.*, 20 (1986), 227-243.

10. IBARAKI, T. AND KATOH, N. *Resource Allocation Problems*. The MIT Press, (1988).

11. IRVING, R. W. AND LEATHER, P. The complexity of counting stable marriages. *SIAM Journal on Computing*, 15 (1986), 655-667.

12. ITOGA, S. Y. A generalization of the stable marriage problem. *Journal of the Operational Research Society*, 32 (1981), 1069-1074.

13. ITOGA, S. Y. A probabilistic version of the stable marriage problem. *BIT*, 23 (1983), 161-169.

14. ITOGA, S. Y. The upper bound for the marriage problem. *Journal of the*

Operational Research Society, 29 (1978), 811-814.

15. KLOP, J. W. Term rewriting systems from Church-Rosser to Knuth-Bendix and beyond. *Automata, Languages and Programming*. Paterson (Ed.), M. S. Springer-Verlag, (1990).

16. KNUTH, D. E. *Mariages Stables*. Les Presses de l'Universite de Montreal, Montreal, (1979).

17. MCVITIE, D. AND WILSON, L. B. Stable marriage assignment for unequal sets. *BIT*, 10 (1970), 295-309.

18. MJELDE, K. M. *Methods of the Allocation of Limited Resources*. John Wiley & Sons, (1983).

19. NEWMAN, M. H. A. On Theories with a Combinatorial Definition of "Equivalence". *Ann. of Math.* 43 (1942), 223-243.

20. RONN, E. NP-complete stable matching problems. *Journal of Algorithms*, 11 (1990), 285-304.