STABLE MATCHING PROBLEM AND AN APPLICATION OF THREE WAY KIDNEY EXCHANGE PROBLEM TO 3-SIDED CYCLIC NETWORKS

Enrollment Number:	122212
Name of the Student:	Kalyani
Name of the Supervisor:	Dr. Nitin

May- 2014

Submitted in partial fulfillment of the Degree of

Master of Technology

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING AND INFORMATION COMMUNICATION TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY WAKNAGHAT

TABLE OF CONTENTS

CERTIFICATE

This is to certify that the work titled **“STABLE MATCHING PROBLEM AND AN APPLICATION OF THREE WAY KIDNEY EXCHANGE PROBLEM TO 3-SIDED CYCLIC NETWORKS”** submitted by **“Kalyani”** in partial fulfillment for the award of degree of Master of technology of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Name of Supervisor : Dr. Nitin

Designation : Associate Professor

Signature of Supervisor:

Date :

ACKNOWLEDGEMENT

This report benefited from the help and advice of many people. Particular thanks go to my supervisor Dr. Nitin who introduced me to stable matching and whose ideas and criticism were crucial to the development of my analytical methodology. His valuable comments and efficient use of time to painstakingly check the correctness of proof of complexity in the preliminary versions of this report helped me a lot. My simulations for the proposed approaches would not have been possible without the help of network administrators of our college who introduced me with the basic workflow of sensor networks and helped me experimentally build one. I would also like to acknowledge Jaypee University of Information Technology for its continual support both financially and technically. Finally, I would like to thank the anonymous referees, for their constructive comments that greatly helped in the improvement of the presentation of this report.

SUMMARY

Three-sided relationship is very common in the social and economic area, e.g., the supplier–firm–buyer relationship, kidney exchange problem. The three-sided relationship can also be found in many scenarios of computer networking systems involving three types of agents, which we regard as three-sided networks. For example, in sensor networks, data are retrieved from data sources (sensors) and forwarded to users through a group of servers. In such three-sided networks, users always prefer to receive the best data services from data sources, data sources would choose servers that are more efficient to deliver their data, and servers try to satisfy more users. Such preferences form a specific cyclic relationship and how to optimally allocate network resources to satisfy preferences of all parties becomes a great challenge. In my work, inspired by the three-sided stable matching, I model the Three-sided Matching with Size and Cyclic preference (TMSC) problem for data sources, servers and users, aiming to find a stable matching for them, where all their preferences are satisfied following a proposed parallel approach ($MOD_{P\text{-}GSA}$) which preserves the merit of basic stable matching by eradicating the occurrence of worst case scenarios and hence are time efficient. TMSC is different from the traditional three-sided matching models, as each server may normally serve more than one users. I have proposed efficient algorithms for the restricted model of TMSC problem to find a stable matching. The effectiveness of the proposed algorithms is validated through lemmas and proofs mathematically and experimentally through extensive simulations.

Signature of the Student Signature of the Supervisor

Name: Kalyani Name: Dr. Nitin

Date: Date:

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

In the field of computer science, we study a lot of things, theoretically, which are not really implementable in the physical world. The concept of binary trees is one of many instances of such non-implementable CSE topics, as always we do not have the things getting recursively divided into two parts. Contrary to this, Stable Matching Problem (SMP), first introduced by two economists David Gale and Lloyd Shapley in the year 1962, provides us with a huge range of real world application areas. The justification of the name comes from the fact that in SMP we make stable pairings/matches between the entities of one set to the other retaining the stability between the matched pair. The root problem SMP, further gave rise to many child problems such as:

(i) College Admissions Problem

(ii) Stable Marriage Problem

(iii) Stable Roommate Problem

(iv) Hospital Resident Problem

(v) Three Way Kidney-Exchange Problem

(vi) Matching inputs/outputs in Switch Schedulers

(vii) Processors/task matching

(viii) Compiler/Register matching and so on.

In the seminal paper, "College Admissions and Stability of Marriage" the economists, David Gale and Shapley considered two variations of SMP; first was College Admission Problem where the task was to assign the students to the colleges according to their prioritized choices keeping in view the priority criteria of the colleges to pick students, in a stable way. The second problem the authors addressed was the Stable Matching Problem, widely known as Stable Marriage Problem which was informally called the matchmaker problem. In a basic matchmaker problem there were a set of n-men and n-women. Each man and woman used to give a strict preference list for the opposite sex without ties and incomplete lists. *Rank* of an entity refers to the position of the entity in the preference list of other. After the matching set/ solution set has been formed the sum of ranks of all the matched entities in each set is known as the *score, S*. Basic GSA states that the score of satisfaction of the requesting entity should be more than that of accepting entity set which is widely known as the *Satisfiability Rule*. The merit of stable matching can only be used efficiently if this rule is followed. In the worst case scenarios, the score of accepting entity set becomes higher than that of the proposing set, which disturbs the harmony of SMP. Later, another variation of SMP came into being i.e. Stable Roommates Problem (SRM) which instead of considering two pools of data, considered only one dataset with each prioritizing its choice for every other member in the pool i.e. if there are n-entities then each will have (n-1) entities strictly ordered in their preference lists. With the emergence of many variations of two sided SMP, another scientist Knuth asked if we can have stable matching with more than two pools of data. To this Ng and Hirschberg came forward with each member specifying the single priority entry as one of the possible combinations of other entities, which was further proved to be NP Complete. Another paper which did not consider making such combinations named the problem to be Circular Stable matching where each party prefers the other in a circular way. SMP also showed its application in the field of medical science addressing the issues like Hospital/Resident Problem and 3-way kidney exchange problem. Hospital/Resident Problem considers the issue where we have different number of members in either pool. Here, number of residents is far more than that of the number of hospitals, and the task is to assign the residents to the hospitals. 3-Way Kidney exchange problem targets making pairs between the patients and the interested donors. The scope of SMP is not only restricted to economic and social areas, but also can be mapped to the field of Computer Science Engineering and Information Communications Technology (CSE & ICT). Some such instances are matching input/output switches in the switch schedulers, processor/tasks matching and Compiler/Register matching. Much research has been done to match the input ports to the output in various interconnection networks to make the overall network fault tolerant by providing with multiple paths for the signal to travel. This in turn increases the performance to a lot extent. Specifically, Multistage Interconnection Networks (MINs) and Gamma Interconnection Networks (GINs) top the list of much researched upon networks. A recent application of stable matching in the 3-sided computer networks exists, but it fails to address the problem in a parallel manner with an aim to provide utmost satisfaction to the end user.

This report summarizes a proposed novel approach: OPTIMAL_NETWORK_MATCH which works on framework for Video-On-Demand in a parallel manner ($GSA_{PARALLEL}$) to match entities from each entity set: users (U), sensors (D) and servers (S) and outputs a stable triplet match such that users attain maximum satisfaction. In this report following objectives have been achieved:

1. An algorithm is proposed to detect and eradicate the worst case scenarios in GSA_{BASIC}.
2. A novel approach, OPTIMAL_NETWORK_MATCH has been proposed to match a group of users to specific servers with an objective to gain utmost satisfaction for users. Experiments are performed on a restricted model for Stable Matching with Incomplete lists and Ties (SMTI). To the best of our knowledge, this is the first one to apply SMTI in a parallel manner to 3-sided cyclic networks and takes care of satisfaction of entities in the direction of request.
3. We have proposed lemmas and theorem to show that OPTIMAL_NETWORK_MATCH stops and for equal number of entities in the user, sensor and server set; it takes maximum n^3 number of steps to complete.
4. Extensive simulations have been done to validate the proposed algorithms by taking basic inputs from the users and servers and generating the preference list dynamically to test on a real network.

This report is divided into 7 chapters. Chapter 2 lays down the literature review for the research done so far. Chapter 3 states the motivation behind the proposed approach. Chapter 4 consists of the basic notations and concepts required with chapter 5 depicting the proposed approach. Chapter 6 pictures the experimental set up and chapter 7 shows the results obtained. Chapter 8 concludes the report.

2. LITERATURE REVIEW

The Stable Matching Problem (SMP) [1] was first introduced by two economists, David Gale and Lloyd Shapley in the year 1962 in their seminal paper "College Admissions and Stability of Marriage". Recently, in 2012 the Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel (Nobel Prize in Economics) was awarded to Lloyd S. Shapley and Alvin E. Roth "for the theory of stable allocations and the practice of market design." This problem is otherwise known as a Matchmaker Problem/ Stable Marriage Problem, matching each man from set1 (say) to a feasible woman of his choice from set2 (say). As when introduced by Gale-Shapley, some assumptions and prerequisites were taken.

a. People are heterosexual i.e. people can set priorities in their preference list for members of opposite sex (set) only.
b. Who will propose whom is already been decided, as it affects the result. The result is more oriented towards the set which proposes rather than the entity set which responds to the proposal.
c. There should be equal number of members in each set. For a stablimagese marriage problem, there should be n-men and n-women
d. Each person is required to specify its preference list for persons from the opposite set.
e. Preference list should be complete (everybody is declared acceptable), and is strictly ordered.

With the above criteria fulfilled we can define *marriage* as a complete 1:1 matching between two sets. We denote the ordering of people in the preference lists by *rank*. Lower is the rank, more is the preference. If a man and woman in different couple prefer each other to their present partner in the match, then it is called as a *blocking pair*. A marriage with no existence of such blocking pairs in known as a *stable marriage*. Figure 1 shows a problem instance for running the *Gale-Shapley Algorithm (GSA)*, with each man and woman specifying a *strict and complete* preference list for the members of the opposite sex.

Problem Statement: Find a stable marriage given the unique preference list of each individual. (1, 2, 3, 4) are men and (A, B, C, D) are women and he assumes the man proposes to woman. Unique preference list is given by:

M_i	Pref_list (M_i)	W_i	Pref_list (W_i)
1	B, D, A,C	A	2, 1, 4, 3
2	C, A, D, B	B	4, 3, 1, 2
3	B, C, A, D	C	1, 4, 3, 2
4	D, A, C, B	D	2, 1, 4, 3

Fig. 1: Basic Problem Instance for GSA

Rules for GSA:

a. All are unengaged at the beginning.
b. While there are unengaged men, each proposes until a woman accepts.
c. Unengaged women accept the first proposal they get.
d. If an engaged woman receives a proposal she likes better, she breaks old engagement and accepts new proposal; dumped man begins proposing where he left off.

Solution:

STEP1: 1→B, B accepts. The matching set, □ is now (1, B).

STEP2: 2→C, as for C it is the first proposal, C accepts and the matching set is updated to (1, B) (2, C).

STEP3: 3→B, in Bs preference list 3>1, B prefers 3 to 1; hence B breaks up with 1 and accepts proposal from 3. Now 1 returns to the set of unengaged men, and the new matching set is given by (3, B), (2, C).

STEP4: 1→D, D accepts the proposal and the matching set becomes (1, D) (2, C) (3, B).

STEP5: 4→D, D denies, as it has already been engaged with a better partner 1, having greater preference than 4, Matching set remains unchanged as (1, D) (2, C) (3, B).

STEP6: Finally, 4→A, A accepts and the final matching set, is now (1, D) (2, C) (3, B) (4, A).

Adding up the ranks of the matched pair for each set we calculate the *happiness* for each entity set. For men, the happiness is given by 6 (2+1+1+2) and for women it is 12 (2+4+3+3). Less is the happiness score more stable the marriage is.

Results:

a. GS always ends up. The worst case complexity is $O(n^2)$.

Case 1- Man proposes to woman!!!

(i) Each man has the best partner he can have.
(ii) Each woman has the worst partner she can have.
(iii) GS produces the same result irrelevant of the order of proposals of men.

Case 2- Woman proposes to man!!!

(i) Each woman has the best partner she can have.
(ii) Each man has the worst partner he can have.
(iii)GS produces the same result irrelevant of the order of proposals of women.

b. Clearly, neither M_m (when man proposes) nor M_w (when woman proposes) are fair enough to both sexes (except when they are the same), so we want some better stable marriage.
c. In the worst case (the happiness score of women is greater than men when man proposes and vice versa), the number of stable marriages grows exponentially with *n*, so exhaustive search is not practical for large *n* (proved by Knuth).
d. The sex equal solution is NP complete.
e. Egalitarian (optimal) solution i.e. treating both sexes equally, can be found in $O(n^3)$ time produces the maximum total happiness.

The results found above pave way to various problems with the basic GSA. Some of them are listed below.

PROBLEM 1: TIES IN THE PREFERENCE LISTS

Ties [2] refers to indecisiveness to lay a strictly ordered list by entities of a set and hence end up specifying same rank to 2/more entities.

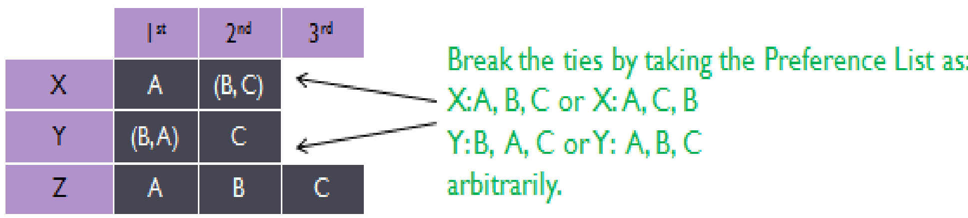


Fig. 2: TIES in preference list

Here, in the above figure X prefers A over B and C, but then becomes indecisive about whether to give B more priority or C. Anyone of them coming second doesn't matter X. For Y he is confused whether to prefer B the most or A, hence has placed both of them in the same rank. The solution to such a problem that has been proposed is to break the ties arbitrarily, and hence achieve weak stability, with no couple strictly preferring each other. But, incase strong stability exists, the algorithm that is adopted to achieve it takes a time complexity of $O(n^4)$ time.

PROBLEM 2: INCOMPLETE PREFERENCE LISTS

The possibility of the preference list remaining incomplete arises basically in two situations. Firstly when there exists some people who may be unwilling to marry some candidates, and secondly when there are uneven number of men and women. In such a case instability occurs in the matching set M if and only if:

a. m_i and w_j do not find each other unacceptable

b. m_i is unmatched or prefers w_j to current fiancé$_j$ is unmatched or prefers m_i to current fiancé

m_i	Pref_list (m_i)		
A	1	3	2
B	2	1	
C	3	2	

Incomplete List

Fig. 3 Incomplete Lists

This problem was further tried to be solved by extending the Gale Shapley Algorithm (GSA) and partitioning candidates into two sets [2]. First set contains those who have partners in all stable marriages, and the second set containing those who do not have any partner in any stable marriage. But this approach may give rise to weakly stable marriage of different size, unlike ties.

PROBLEM 3: MAXIMUM CARDINALITY WITH TIES AND INCOMPLETE LISTS (MAX-SMTI)

With Incomplete Lists and Ties both co-existing in a problem instance here, an open research problem arises as how to have a stable matching done in such a case? Till now research has been done to solve this using network flow diagrams, but the problem becomes NP Hard when we try finding weakly stable marriage of maximum cardinality, even if it's the case that only women declare ties.

Irving and Manlove in 2004 [] proposed an algorithm to solve MAX-SMTI with a time complexity of $O(2-c(\log n)/n)$ which has been later improvised by Kazoo Iwama, Shuichi Miyazaki, Naoya Yamauchi in 2008 by $O(2-c/\sqrt{n})$. This algorithm considers arbitrary initial table match and at each stage it improves the size of match. For, |M| be the size is current solution, in each step to increase the size of matching we remove H subset of M and compute K having some good property (say, β) such that K = 2 |H|. This approach proposes an efficient approach instead of naïve exhaustive search by Irving, to find H which will take time less than *logn*. LOCALSEARCH(l) proposed by Irving follows an exhausite search method which has been replace by INCREASE to obtain a better upper

MaxMatching*(M)*

1. Construct a bipartite graph $G = (U,V,E)$ in the following way: $U = MX$. V is the set of men not in M_Y. Include an edge between $w \in U$ and $m \in V$ if and only if $M(w) =_w m$.
2. Find a maximum matching in G, and output it.

MultipleGS*(M, S)*

1. Let $Y = S^Y$.
2. Let X = the set of women not in M^X.
3. **For each** man $m \in Y$, do the following: Delete all women not in X from m's preference list. Break all ties in m's preference list in an arbitrary way.
4. **End For**
5. **For each** woman $w \in X$, do the following: Delete all men not in Y from w's preference list. Break all ties in w's preference list in an arbitrary way.
6. **End For**
7. **For** $k = 1$ to $2B - 1$, do the following:
8. **For each** man $m \in Y$, delete, from m's preference list, all women whose position (with respect to m's modified list) is greater than k. Apply men-propose GS algorithm to Y and X, and let the resulting matching be Q_k.
9. Partition Y into $|Y|/2B$ sets $Y1, Y2, \ldots, Y|Y|/2B$ arbitrarily, each of which is of size $2B$.
10. For each Yi, do the following: For each man $m \in Yi$, delete, from m's preference list, all women whose position (with respect to m's modified list) is greater than $2B$. Apply men-propose GS algorithm to Yi and X, and let the resulting matching be Q_{2B-1+i}.
11. Output $Q1, \ldots, Q2B-1+|S|/2B$.

INCREASE(*M*)

1. **Begin**
2. Execute MaxMatching*(M)* and obtain a matching P.
3. Let M' be the subset of M such that $w \in M'^x$ *iff* $w \in P^X$.
4. Execute MultipleGS*(M,M')* and obtain $Q1, \ldots, Q2B-1+|M'|/2B$.
5. **For each** Qi, do the following.
6.

   $Mi = M - M_i''$.

   For each woman $w \in M_i''^X$, add an

   Let M_i'' be the subset of M' such that $m \in M_i''Y$ iff $m \in Qi^y$.

   For each man $m \in M_i''^Y$, add an edge $(m,Qi(m))$ to Mi.

   Construct a bipartite graph $Gi = (Ui,Vi,Ei)$ as follows: $Ui = M_iY$ and $Vi = M_iX$. Include an edge between $m \in Ui$ and $w \in Vi$ iff (m,w) blocks Mi.

   End For
7. Find a minimum vertex cover Ci for Gi.
8. From Mi, delete all edges connected to at least one vertex in Ci.
9. **For each** i, let $M1,i = Mi$.}

10-16. Do the same operation as lines 1 through 9 by exchanging the role of men and women, and let the resulting matchings be $M2,i$.

17. **End For**

18. Let M^* be a largest matching of all $M1,i$ and $M2,i$.

19. If $|M^*| > |M|$, output M^*. Otherwise, output failure.

20. **End For**

bound, and it shows that it never fails if $|M| < OPT/2 + c'\sqrt{|M|}$. Here c' is a positive constant such that $c' <= 1/8\sqrt{6}$. Hence, LOCALSEARCH can obtain a stable matching of size at least $OPT/2 + c`\sqrt{|M|}$. *Maxmatching(M)* takes a matching set, M as parameter and outputs a matching that matches a subset of women in M with a subset of unengaged/single men in M. *MultipleGS* finds several possible matchings between *Y* and *X* using man-oriented GSA. *INCREASE* computes for all possible values of *i*. Fig. 4 shows the computation of INCREASE function over a fixed value of *i*. Fig. 4(1) shows an input matching M. M is given to the MaxMatching procedure which returns P, in Fig. 4(2). This P computes M', and then M and M' are passed to MultipleGS function. Figure 4(3) shows Y and X used in the function MultipleGS. Out of many matchings returned by MultipleGS, one matching Q_i is fixed in Figure 4(4). Then, M''_I is computeded by this *Qi*.

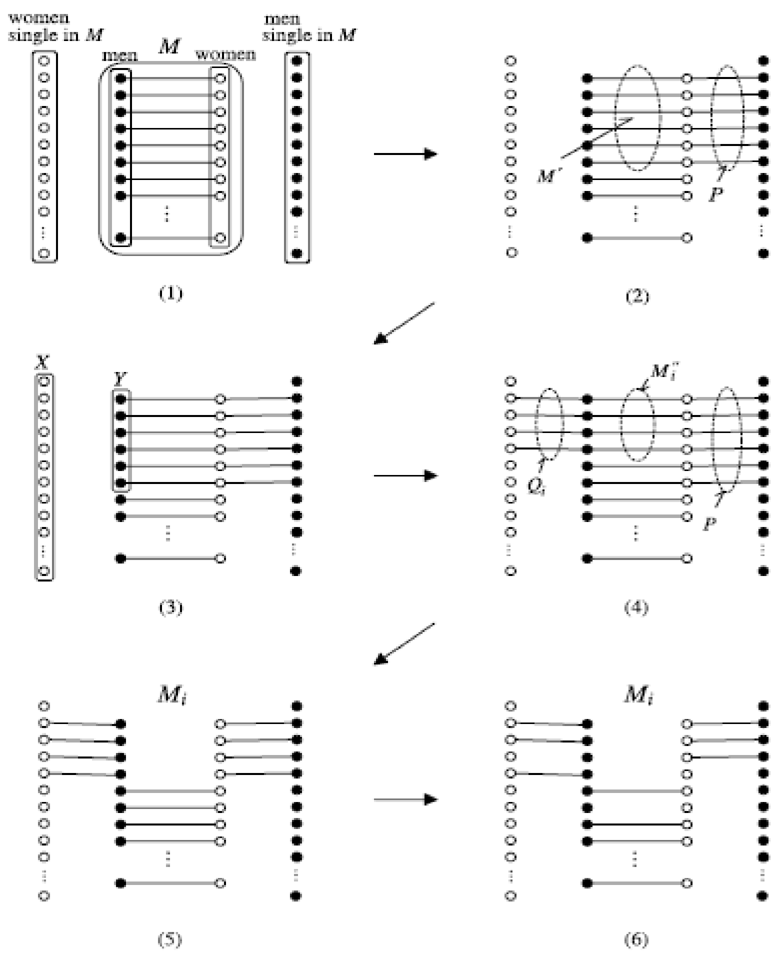


Fig. 4 An example of computation by INCREASE

Next, INCREASE procedure increases the size of M by removing edges *(m,w)* from M_i" from M, and by matching each m and w to a single woman and man, respectively, according to P and Qi . The resulting matching Mi is shown in Fig. 4(5). However, Mi may break the property β. At lines 9 through 11, it removes some edges of Mi so that the matching satisfies β in Fig. 4(6). This decreases the size of matching.

PROBLEM 4: CYCLE/DEADLOCK FORMATION WITH BLOCKING PAIRS

A cycle/deadlock [4] is formed when in case of a blocking pair, pairs divorce and the divorced partners marry each other leading to a cycle.

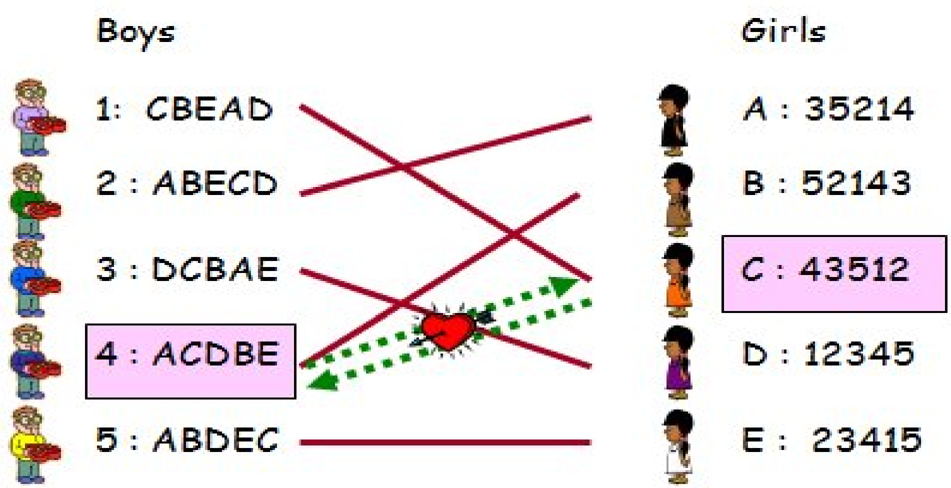


Fig. 5: Blocking Pairs leading to cycle formation

In the above figure, both 4 and C prefer to each other than their currently assigned partners. One solution to unblock this is for 4 and C to break up with their current partners B and 1 respectively and elope together. But in this case, problem arises when B and 1 marry each other. This leads to formation of a cycle and hence a deadlock.

PROBLEM 5: CHEATING BY PLAYERS

If any player [5] cheats by specifying a deceitful preference list to get his/her desired partner. If one man lies and others are true about their preferences then man cannot cheat, though the woman can cheat arranging the men in her preference list in a deceitful manner. Cheating is desirable on the women side as it's always a man oriented approach when man proposes. An example of such an instance is shown in the figure below. The way woman can cheat to get her man of dreams is now explained. At first she remains blank declaring any of the men un-acceptable for her. Then after knowing the order of proposal of men she can specify which man she wants to be paired up with and can accordingly specify her preference list stating all men as unacceptable after the one she wants to tie knot with. This makes her get that man in order to have the whole marriage stable. But in some cases, woman is not allowed to keep the men unacceptable.

	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Men's Preference List

	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Women's True Preference Profile

	1st	2nd	3rd
A	Y	Z	X
B	X	Y	Z
C	X	Y	Z

A Lies

Fig. 6: Cheating by women

A natural extension suggests woman' w' to:

a. Accept a proposal, and then reject all future proposals.
b. From the list of men who proposed to *w* but were rejected, find her most preferred partner; repeat the Gale-Shapley algorithm until the stage when this man proposes to her.
c. Reverse the earlier decision and accept the proposal from this most preferred partner, and continue the Gale-Shapley algorithm by rejecting all future proposals.
d. Repeat (b) and (c) until the woman cannot find a better partner from all other proposals.

DISADVANTAGE: The above strategy does not always yield the best stable partner a woman can achieve. The reason is that this greedy improvement technique does not allow for the possibility of rejecting the current best partner, in the hope that this rejection will trigger. To solve this issue, some authors later described one more approach, but it was not even free from anomalies.

2.1 VARIANTS OF STABLE MATCHING PROBLEM

2.1.1 PARALLEL STABLE MATCHING

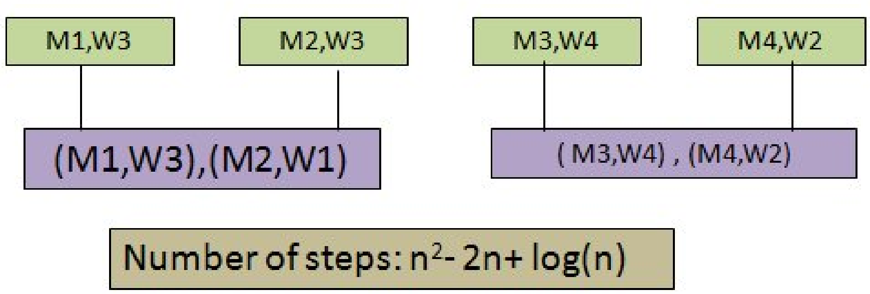
In the basic stable matching algorithm, considering a man-oriented approach, while each engagement of man, who is engaged more than once, is less desirable to him, the woman gets successively more favorable engagements. At most the rejections for each man are (n-1) and the last woman doesn't make any rejection at all. Hence, the worst case scenario has (n-1)(n-1) rejections making it as $O(n^2)$. A man never proposes to the same woman more than once. Some researchers tried to decrease this complexity by following a parallel approach [6] instead of sequential one. [7] proposed two parallel algorithms for stable marriage problem implemented on a MIMD parallel computer. They took random and worst case instances are taken into consideration. The normal parallel approaches made by Tseng and Lee proceeds through divide and conquer principle. Having an instance P of the stable matching problem, P is in the dividing step is divided into2 sub-problems as P1 and P2. P1 contains men from (1 to n/2) and P2 contains men from (n/2 + 1 to n). Each sub-problem is stable locally, and the conflicts in the sub-problems with a single man assigned to 2 women are solved by seeing the women's preference list. This thing is carried out recursively to solve the whole problem.

	1st	2nd	3rd	4th
m1	3	2	1	4
m2	3	1	2	4
m3	4	3	1	2
m4	2	4	3	1

Men's Preference Profile

	1st	2nd	3rd	4th
w1	1	3	2	4
w2	4	1	3	2
w3	4	3	1	2
w4	2	4	3	1

Women's Preference Profile



Number of steps: $n^2 - 2n + \log(n)$

Fig. 7 Parallel Gale-Shapley Algorithm

This approach when executed as such without any parallel processors included proved to take more time than the sequential ones. Considering, MEIKO parallel computer, which are message passing, distributed memory systems. They contain 16 intel i860 processors, each with 16Mb memory, 32 T800 communication transputers. One main disadvantage with MEIKO computers is that, they have high start-up latency, which leads to even communication of small amounts of data, expensive in time. For executing the parallel algorithm [8] have divided it into two phases: Phase I is known as the *proposing phase* where all free men propose to the woman highest on their preference list simultaneously. In phase II i.e. the *rejection phase*, the women in parallel evaluate their proposals keeping the best of the proposal and rejecting the remaining men. Now free men again propose to the second woman on the preference list and the process continues. As the MEIKO computers had high start-up latency vs. integer operation ratio, need is to keep the communication low. One probable solution this, as identified by [8] is the *master slave approach*. The master deals with the phase I and slave the phase II. Master is in charge of the dataflow of the whole algorithm. The master packs the proposals and sends them to the appropriate slave processors thereby minimizing the number of communications. Each woman returns the rejected men back to master in an array. When no man is returned to the master the algorithm terminates. But the main loophole in this is the master. Now having the normal parallel approach be done with the MEIKO processors naming it as *smart parallel approach*, we assign the root processors the main problem which is then further divided into 2 sub-problems and are sent to the k/2 left and right processors. Here, idling can be a problem which they have eradicated by letting more processes run on the same processor, such as having the left child to problem node on the same processor as the parent and the right child on a different processor. But this way we need to keep the problem size small as the memory in one processor gets shared between the problems. Another restriction in parallel approach is that this algorithm only runs for complete binary trees (1, 2, 4, 8, 16 etc.). Even perfect division of k/2 processors can be a bottleneck, which they have eradicated with load balancing. Another factor while balancing the loads we need to consider is that, the problem when subdivided goes first to the left child and then to the right child, which lets the left child have an edge over the right one finishing the matching sooner than its counter-part. Hence what we need to do is to assign the left child a bit more problems than the right child for all sub-problems.

ALGORITHM A

(An algorithm which produces a male optimal stable solution)

Input: A male ranking matrix and a female ranking matrix.

Output: A male optimal stable solution.

Step 1: Divide the problem into two sub-problems, by halving the male ranking matrix. Call these two sub-problems P1 and P2.

Step2: Recursively apply this algorithm to find male optimal stable solutions for P1 and P2- Call these two solutions S1 and S2.

Step3: Apply Algorithm B which is a merging algorithm to combine S1 and S2 into S.

In this algorithm, a merging procedure is used. Let us first introduce some definitions. In a solution, suppose $Mi_1 .... Mi_k$, $k >= 2$, propose to the same woman Wi. Without loss of generality, we shall assume that so far as Wi is concerned, the ranking of Mi_k is the highest, i.e. it will accept Mi_k and will reject m1…..Mi_{k-1}. We shall say that {Mi_1, ... Mi_{k-1} } is the set of rejected men of Wi. In a solution S, let Wi denote the set of women who are proposed to by more than one man. Let R_s be the union of the sets of rejected men of members in W_s. Then R_s is called the set of rejected men associated with S.

ALGORITHM B

(A merging algorithm which produces a male optimal stable solution out of two male optimal stable sub-solutions)

Input: Two male optimal stable solutions S1 and S2 and their associated ranking matrices.

Output: A male optimal stable solution which combines S_1 and S_2.

Step 1: Let S be the union of S_1 and S_2.

Step 2: If no two men propose to the same woman in S, then accept S as the solution and return. Otherwise, go to Step 3.

Step 3: For each man M_i in R_s, and for the set of rejected men associated with S, replace (M_i, W_j) in S by (M_i, W_k) where W_k is the next best choice of M_i.

Step 4: Go to step 2.

When the research was done based upon the above algorithms, the following results were concluded:

a. For a worst case execution of McVitie-Wilson's algorithm the following two statements are true:

   STATEMENT 1: Let M_k finally propose to this N^{th} choice W_i. Then the Nth column of the male ranking matrix consists only of one woman W_i and the $(N-1)^{th}$ column consists of all the other N-1 women.

   STATEMENT 2: Let M_j propose to his $(N - 1)^{th}$ choice W_a. Then the first choice of W_a must be M_j.

b. If $N^2 - N + 1$ proposals are needed in McVitie-Witson's algorithm to obtain the male optimal stable solution, then the probability that this worst case occurs is of the order $CN^{-2N+7/2}e^{-N}$ with $2.5 < C < 3.5$.
c. The number of steps needed to obtain the male optimal stable solution by using our parallel algorithm is at most$N^2-2N+\text{floor}(\log_2 N)$.
d. In the worst case of our parallel algorithm the first column of the male ranking matrix consists of exactly N-1 women.
e. The probability that the worst case occurs in parallel algorithm is less than $CN^{-2N+5}e^{-2N}$ with $30<C<70$, for $N>2$.

CONCLUSION: Parallel algorithms are slower than the sequential ones and they get even slower as the number of processors increase if no special processors are used. Communication is expensive also.

2.1.2 DYNAMIC MATCHING

In the real world, men and women can enter and leave the market. Even men can set/ change their preferences arbitrarily (requires market updating). Therefore, in such a scenario, as proposed by [9], constraint needs to be imposed as we can switch from one matching to another only if there is consensus among the applicants to agree to the switch. A *voting path* here is defined as sequence of matching (each more popular than its predecessor). The time complexity to find the shortest length voting path is given by T(n) and to find a popular path time complexity, T(n) is given by O(n+m) and more generally $O(m\sqrt{n})$.

Here, we say an applicant a prefers matching M’ to M if

a. a is matched in M’ and unmatched in M, or
b. a is matched in both M’ and M, and a prefers M’(a) to M(a).

If most of the nodes prefer M’ to M then we say that M’ is more popular than M, denoted by M’>M. A popular matching doesn’t exist always, as the “more popular than” relation doesn’t exists always. Probability for a dynamic path to exist increases if

a. No of women a slight greater than the multiplicative factor of the number of men.
b. Preference lists are randomly constructed.

For, problem instance

m_i	**Pref_list (m_i)**		
m_1	w_1	w_2	w_5
m_2	w_3	w_2	
m_3	w_3	w_2	
m_4	w_1	w_4	

Let,

$\square_0 = \{(m_1, w_5), (m_2, w_2), (m_3, w_3), (m_4, w_1)\}$

$\square_1 = \{(m_1, w_2), (m_2, w_3), (m_4, w_1)\}$

The only popular matching are $\square^* = \{(m_1, w_1), (m_2, w_2), (m_3, w_3), (m_4, w_4)\}$, and $n^* = \{(m_1, w_1), (m_2, w_3), (m_3, w_2), (m_4, w_4)\}$. Among $(\square_0, \square_1)$, $(\square_0, \square^*)$ and $(\square_0, n^*)$ *voting path* for each is 2.

2.1.3 DISTRIBUTED MATCHING

In the classical case i.e. GSA_{BASIC} each person has to follow a rigid role, making public his/her preferences to achieve a global solution. But, each person may desire to act independently, by keeping private his/her own preferences. However, this problem is very suitable to be treated in a distributed way, trying to provide more autonomy to each person, and to keep preferences private, thus enforcing privacy. The Extended Gale-Shapley algorithm (*EGS*) is an extended version of it that avoids some extra steps by deleting from the preference lists certain pairs that cannot belong to a stable matching [10]. Say, for instance we have the following problem instance as shown in Fig. 8. Upon this instance we will run

m_i	Pref_list (m_i)			w_i	Pref_list (w_i)		
m_1	w_2	w_3	w_1	w_1	m_1	m_2	m_3
m_2	w_1	w_2	w_3	w_2	m_1	m_3	m_2
m_3	w_2	w_1	w_3	w_3	m_2	m_1	m_3

DISTRIBUTED MATCHING

1. Assign each person to be free;
2. **While** some man m is free and m has a nonempty list loop
3. w = first woman on m's list; f_m proposes to w_g
4. **If** m is not on w's preference list then
5. Delete w from m's preference list;
6. Goto line 3
7. **End If**
8. If some man p is engaged to w then
9. Assign p to be free;
10. **End if**
11. Assign m and w to be engaged to each other;
12. **For** each successor p of m on w's list loop
13. Delete p from w's list;
14. Delete w from p's list;
15. **End loop;**
16. **End loop;**

Fig.8 Problem Instance for distributed matching along with the EGS algorithm

During execution of *EGS*, some people are deleted from preference lists. The reduced preference lists that resulted from applying man-oriented Gale-Shapley Algorithm (GSA) are called *man-oriented Gale-Shapley lists* or *MGS-lists*. On termination of algorithm, each man is engaged to the first woman in his (reduced) list, and each woman to the last man in hers. These engaged pairs in the match, □ constitute a stable matching, and it is called *man-optimal* (or *woman-pessimal*) stable matching since there is not other stable matching where a man can achieve a better partner (according to his ranking). Similarly, exchanging the role of men and women in *EGS* with women taking the initiative to propose, we obtain the *woman-oriented Gale-Shapley lists* or *WGS-lists*. On termination, each woman is engaged to the first man in her reduced list, and each man to the last woman in his. These matched pairs constitute a stable matching, and it is called *woman-optimal* (or *man-pessimal*) stable matching. The intersection of both the lists i.e. *MGS-lists* and *WGS-lists* is known as the Gale-Shapley lists (*GS-lists*).

m_i	PL (m_i)	w_i	PL (w_i)
m_1	w_2	w_1	m_2
m_2	w_1	w_2	m_1
m_3	w_3	w_3	m_3

Fig. 9 GS List

These lists have important properties:

a. all the stable matchings are contained in the *GS-lists*,
b. in the *man-optimal* (*woman-optimal*), each man is partnered by the first (last) woman on his *GS-list*, and each woman by the last (first) man on hers.

The *EGS* algorithm that solves the classical *Stable Matching with Incomplete Lists (SMI)* can be easily adapted to deal with the distributed case. We call this new version as Distributed Extended Gale/Shapley (*DisEGS*) algorithm. As in the GSA_{BASIC} case, the *DisEGS* algorithm has two phases, the man-oriented and the woman-oriented, which are executed one after the other. Each phase produces reduced preference lists for each set. The intersection of these lists produces a *GS list* per person. As in the GSA_{BASIC} case, the matching obtained after executing the man-oriented phase is a stable matching (*man-optimal*).

DIS_EGS_MAN()

```
m ← free;
end ← false;
while ~end do
if m = free and list(m) !=φ ; then
w ← first(list(m));
sendMsg(propose,m,w);
m ← w;
msg ← getMsg();
switch msg:type
accept : do nothing;
delete : list(m) ← list(m) - msg.sender;
if msg:sender = w then m ← free;
stop : end ← true;
```

DIS_EGS_WOMAN()

```
w ← free;
end ← false;
while ~end do
msg ← getMsg();
switch msg.type
propose: m ← msg.sender;
if m ∉ list(w) then
sendMsg(delete,w,m);
else
sendMsg(accept,w,m);
w ← m;
for each p after m in list(w) do
sendMsg(delete,w,p);
list(w) ← list(w) ¡ p;
stop : end ← true;
```

Fig. 10 Distributed man oriented and woman-oriented approach

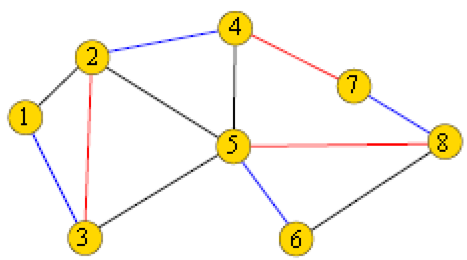
The following messages are exchanged:

a. propose: *m* sends this message to *w* to propose match;
b. accept: *w* sends this message to *m* after receiving a propose message to notify acceptance for match;
c. delete: *w* sends this message to *m* to notify that *w* is not available for *m*; this occurs either (i) proposing *m* an engagement to *w* but *w* has a better partner or (ii) *w* accepted an engagement with other man more preferred than *m*;
d. stop: this is an special message to notify that execution must terminate; it is sent by an special agent after detecting quiescence.

DisEGS algorithm guarantees privacy in preferences of agents and in the final assignment, each person knows the assigned person and no person knows more than that. In this case, it is a kind of ideal algorithm because it assures privacy in values and constraints.

2.1.4. RANDOM MATCHING

Bipartiteness of graph is crucial for the solvability of a matching problem. The GSA_{BASIC} for example does not work for non-bipartite problems like the stable roommates problem. In fact some researchers did believe that the roommates problem was NP-complete, but more than 20 years after the Gale-Shapley paper Robert Irving presented a polynomial time algorithm for the stable roommates problem [10]. According to Irving's algorithm, it either outputs a stable solution or "No" if none exists. This was a major breakthrough achievement, but still the problem was (and is) not fully understood, see e.g. the "Open Problems" section in [11]. One of the open issues being the probability *Pn* that an arbitrary roommates instance of size n is solvable. Numerical computations indicate that *Pn* is a monotonically decreasing function of n, but the data ". . . is not really conclusive enough to add support to any strong conjecture as to the ultimate behavior of *Pn*". In this contribution, the authors of [11] have presented numerical data that *is* conclusive enough to conjecture the asymptotic behavior of *Pn*.



1:	2	3			
2:	3	1	5	4	
3:	2	5	1		
4:	5	2	7		
5:	3	2	8	4	6
6:	5	8			
7:	8	4			
8:	5	7	6		

Fig. 11 Example of a stable matching problem: acceptability graph *G* (left) and preference table *T* (right). The matching indicated by blue edges covers all vertices but is not stable. The red edges form a stable matching.

It is well-known that not all problem instances on non-bipartite graphs (for example an odd cycle) admit a stable matching. They have presented numerical results for the probability that a graph with n vertices and random preference relations admits a stable matching. Findings state that that this probability decays algebraically on graphs with connectivity Q(n) and exponentially on regular grids. On finite connectivity, Erd¨os-R´enyi graphs the probability converges to a value larger than zero. Based on the mathematical results and some heuristic reasoning Stephan Mertens et al. have formulated five conjectures on the asymptotic properties of random stable matchings.

2.2 Applications of Stable Matching Problem

NATIONAL RESIDENT MATCHING PROBLEM: The introduction of Stable Matching Problem (SMP) by Gale and Shapley was conceptualized by the matching between medical students and hospitals in the US, currently known as NRMP (National Resident Matching Program) [12]. There are three other students-hospitals matching systems that use SM (more precisely, the hospitals/residents problem, introduced in Sec. 6.2); CaRMS [9] in Canada, SPA [13] in Scotland, and JRMP [14] in Japan.

MAN EXCHANGE STABLE MARRIAGE For the GSA_{BASIC}, a new stability definition, *man-exchange stability* was defined. This stability requires, in addition to the basic stability, the property that no two men prefer to exchange their partners. Irving et al have proved that the problem of asking the existence of a man-exchange stable matching is NPcomplete [15].

MANY-TO-MANY STABLE MARRIAGE We can consider more of original variant than Hospital Resident, a many-to-many extension of the stable marriage, so that both men and women have quota. One may easily see that the copying technique used in reducing HR to table matching cannot be applied any more since if we do so, the resulting stable matching may create multiple copies of the same pair. Baiou and Balinski [16] showed that some properties for one-to-one and many-to-one case of stable matching also hold for this case. Bansal et al. [17] gave an efficient algorithm for finding a minimum egalitarian stable matching in this scenario. Malhotra [18] studied many-to-many stable marriage with ties, giving an efficient algorithm for finding a strong stable matching, and also proving that all strong stable matchings form a distributive lattice.

STUDENT-PROJECT ALLOCATION PROBLEM *Student-Project Allocation Problem* (*SPA*) is a variant of Hospital resident Problem, in which students are assigned to projects based on their preferences for projects. One lecturer may provide two/more projects, and in that case, all projects that are given by the same lecturer have the same preference list forming a tie. Each project has its own quota i.e. number of students who can take up this, and each lecturer also has his/her quota. We are asked to find a stable matching that satisfies all quota-constraints and preference lists both for projects and lecturers. Abraham et al. [19] gave two algorithms to tackle this problem and also studied its structural properties.

3-DIMENSIONAL MATCHING The 3-DSM was proposed by Knuth [20], in which we are given three sets of agents. There exist a range of freedom in modeling this problem, such as the form of preference lists and the stability definitions. Ng and Hirschberg [21] and Subramanian [22] proposed one model and proved the result to be NP-complete. The complexity of another model, called *cyclic 3D stable matching*, is a open research problem but partial results can be found in Boros et al. [23] and Eriksson et al. [24].Recently, 3-dimensional table roommate was studied by many groups [25 - 27].

ONE-SIDED PREFERENCE LIST There are some matching problems in which only one party (say, men) needs to specify preference lists over the other. A *rank-maximal matching* (or a *greedy matching*) is a matching that matches the maximum number of men to their first preferred partners, and subject to this condition, the maximum number of men to their second ranked partners, and so on. The problem of finding a rank-maximal matching was studied by Irving [28] and Irving et al. [29], who gave polynomial time algorithms. For two matchings $M1$ and $M2$, if the number of men who prefer $M1$ over $M2$ (in terms of the rank of his partner) is greater than that of men who prefer $M2$ over $M1$, we can say that $M1$ is *more popular than* $M2$. A matching M is *popular* if there is no such matching more popular than M. Abraham et al. gave a polynomial time algorithm to decide whether a given instance admits a popular matching, and finds a largest one if any [30]. Mestre [31], and Manlove and Sng [32] solved the weighted version and many-to-one version of this problem, respectively.

STUDENT ADMISSIONS AND FACULTY RECRUTMENT PROBLEM In these types of problems students or faculties specify preference list for universities and vice versa. Being a variation of many-to-one stable matching problem, these problems are modelled and analyzed in terms of graphs. Stable assignments/ matches, which are potentially exponential in number, form a distributive lattice whose *sup* and *inf* are the applicant-optimal and university stable matches, $\square_A$ and $\square_V$ respectively.

STABLE ROOMMATES PROBLEM Stable roommate [33] is the generalized view of non-bipartite graph of stable matching problem. Here there is no disjoint set of entities i.e. it does not require to categorize entities into two set (in general, set of men and set of women). It is different from stable marriage problem as any entity can prefer any other entity in the same set. Further it was also proved that stable matching for the instance of stable roommate may not exit, and answer to whether such a solution exists or not in NP Complete. SR blocking pair is also defined in terms of three notations i.e. Weakly Stable, Strong Stable, Super Stable as discussed in earlier section of Stable marriage problem. As already discussed there is no guarantee of finding stable matching in SR without ties. Hence breaking ties in SR does not guarantee that stable matching will be reported. Ronn [34] proved that the problem of resolving whether SRT instance admits a weakly stable is NP-complete. Further Irving and Manlove [35][36] showed the decision on weak stability in SRT and SRTI and also proved weak stable matching instances of SRT may have distinct sizes.

Scott [37] presented O(a^2) algorithm for finding strong stable matching for SRTI instances where a is the length of the preference list. Algorithm also processes in two- phases and is somewhat complex than super stable matching.Given a problem instance as shown in Fig. 12, the working of SRM is described below.

P_i	**Pref_list (P_i)**				
P_1	P_3	P_4	P_2	P_5	P_6
P_2	P_6	P_5	P_4	P_1	P_3
P_3	P_2	P_4	P_5	P_1	P_6
P_4	P_5	P_2	P_3	P_6	P_1
P_5	P_3	P_1	P_2	P_4	P_6
P_6	P_5	P_1	P_3	P_4	P_2

Fig. 12 Problem Instance for Stable Roommates Problem

Say, we have the matching set, □ contains the pairs {(p_1, p_4), (p_2, p_6), (p_3, p_2), (p_4, p_5), (p_5, p_3), (p_6, p_1)} is then □ will be blocked by following pair {(p_6, p_1); (p_2, p_6); (p_1, p_4)}. Knuth theoretical proved that problem of finding the existence of stable matching is a NP-

Complete problem. Later, Irving [38] provided a polynomial time algorithm that finds the existence of stable match if does not exist. Irving's algorithm consists of two distinct phase:

Phase1: This phase is similar to GSA_{BASIC} where each participant proposes to other based on his/her preference list or continues to propose next person if current proposal is rejected. A participant rejects the proposal if he/she is already holding a proposal or respectively receive proposal from one he/she prefers the more. An important observation was made in this phase that one participant might be rejected by all of the others person thereby proving that no stable matching exists or ends with proposal hold by each person. On the termination of phase1, SRM's reduced preference list is generated known as Stable Table or Phase1 table. After phase 1 execution Fig. 12 ends with reduced preference list as shown in Fig 13 where 1st preference shows the sequence of proposal and deletion shows rejections.

STABLE ROOMMATES PHASE I

Input: Stable Roommate Instance

Output: Stable Table

```
Assign each person to be free;
While some free person Pi has a non-empty list do
    Pj:= first person on Pi's list;
    If some person Pk is semi-assigned to Pj then
        Assign Pk to be free;
        Assign Pi to be semi-assigned to Pj ;
        For each successor Pl of Pi on Pj's list do
            delete the pair (Pl, Pj);
        End for
    End if
End while
```

P_i	Pref_list (P_i)				
P_1	P_4	P_2	P_5	P_6	
P_2	P_6	P_5	P_4	P_1	P_3
P_3	P_2	P_4	P_5		
P_4	P_5	P_2	P_3	P_6	P_1
P_5	P_3	P_2	P_4		
P_6	P_1	P_4	P_2		

Fig. 13 Algorithm and Reduced Table for SRM Phase I

Phase 2: In this phase the preference list is reduced by eliminating sequence of rotations. This phase continues until each person's preference lists have one person indicating stable match have been found or participant's list becomes empty indicating no stable matching exists for such instance.

```
STABLE ROOMMATES PHASE II
Input: Stable Table Instance
Output: Stable Matching M
T = Stable Table;
    While (T list is non-empty) do
        Identify rotation R in T;
        Eliminate R from T;
        If (T list becomes empty)
            Return Null; // no stable matching exit
        Else If (reduced List become of size 1)
            Return Stable –Pair (x,y); // found the stable pair
        End if
    End while
```

Fig. 14 Algorithm for SRM Phase II

Using stable table in Fig. 13, rotation R1 is identified i.e. (P_1, P_4), (P_3, P_2) as P_2 is P_1's second preferred person and P_4 is second ranked person of P_3. Thus we need to remove R1. Next, the rotation R2 = {(P_1, P_2), (P_2, P_6), (P_4, P_5)} is identified. After eliminating, stable match found is P_1 and P_6. At end of phase 2 stable matching found is {(P_1, P_6), (P_2, P_4),(P_3, P_5)}. Irving proved that stable roommate algorithm terminate in $O(n^2)$ and also stated that given SM instance we can construct SR instance such that stable matching is one–to–one correspondence. Generalization of stable roommates includes considering Ties and Incomplete List i.e. Stable Roommates with Ties and Incomplete Lists (SRTI).

STABLE MATCHING IN BIPARTITE GRAPHS An instance of stable marriage is considered as bipartite graph, *G=(XUW,E)*, where *G* denotes the graph with *X* , *W* vertices and *E* edges. Adjacency lists are linearly ordered with ties being allowed. Considering *n* as no. of nodes (men/women) and *m* as no. of edges an edge can be denoted as (a, b). In case *(a, b)* and *(a, b')* are tied, we say that *a* is indifferent between *b* and *b'*. If edge (a, b) preceeds (a, b'), then we can say that a prefers b to b'. Stable matching problem is said to be complete, if number of men equals to number of women and hence, G is complete bipartite graph with $m = (n/2)^2$. In a graph, a matching □ is a set of edges where no two of which share an end-point. If (a, b) Є □ we say that a and b are partners. An edge E = (a, b) Є E\M is a blocking pair if a is unmatched/ a strictly prefers b to his or her currently matched partner in M and b is unmatched/ b strictly prefers a to his or her currently matched partner or is indifferent between them i.e. if a prefers to match with b, b would not object. Blocking pairs can the solved locally by divorcing the current partners and marrying each other, but it forms a cycle if the divorced partners marry each other (though the chances are rare). Iriving in 1994 showed that the time complexity, for computing strongly stable matching in complete instances is $O(n^4)$. In 1999, Manlove[39] showed it for incomplete bipartite graphs to be $O(m^2)$. Later in 2003, [40] an improved Iriving algorithm was proposed taking O(mn) time.

m_i	Pref_list(m_i)		w_i	Pref_list(w_i)	
m_1	w_1	w_2	w_1	m_2	m_1
m_2	w_1, w_2		w_2	m_2	m_1

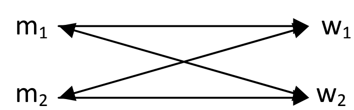


Fig. 15 Problem Instance for SMP in graphs

Here both women prefer to m_2 over m_1. m_1 prefers w_1 over w_2 and m_2 is indifferent between women, w_1 and w_2. The matching $\{(m_1, w_1), (m_2, w_2)\}$ is not strongly stable since, w_1 prefers to m_2 over m_1 and m_2 is indifferent between w_1 and w_2.

3-SIDED CYCLIC NETWORKS Knuth proposed the concept of 3-way stable matching in the year 1976 in [41] with three types of agents men-women-dogs, popularly known as three gender SMP. Matching-set is formed of disjoint families, in the form of triples. Such a triplet is known to be stable, if no blocking family exists i.e. preferred to by all members in current families in a match. In 1988, Alkan [42] gave an instance where no stable matching exists in 3DSM, which was further filtered by Ng and Hirschberg in 1991 and it was shown that the problem of deciding the existence of solution for 3DSM is NP Complete. In 1994, an alternative proof was given by Subramanium [43]. In 2002, Manlove proposed a solution to find SMP of maximum cardinality for an instance of SMTI, and proved that in some cases it is NP Hard. Later, in 2004 Boros et al proved that for *n* equals to 3 stable matching always exists, which was further extended for n equals to 4 by Eriksson et al in 2006. Huang in 2007 [44] showed that the problem remains NP complete for strong stability even if the preference list is consistent. A preference list is inconsistent if, for example, man m ranks (w1, d1) higher than (w2, d1), but he also ranks (w2, d2) higher than (w1, d2), so he does not consistently prefer woman w1 to woman w2. Structural and #P completeness results for string stable matchings. Later in 2009, Peter brio and Eric McDermid [45] showed that given a stability criterion, strong stability (there exists no weakly blocking family), the problem is NP complete even for complete lists. Taking incomplete lists into consideration without occurrence of ties, Fig. 15 shows that no stable matching exists where n equals 6.

$m_1 : w_1, w_1'$	$w_1 : d_1, d_1'$	$d_1 : m_2, m_2'$
$m_2 : w_2, w_2'$	$w_2 : d_2, d_2'$	$d_2 : m_3, m_3'$
$m_3 : w_3, w_3'$	$w_3 : d_3, d_3'$	$d_3 : m_1, m_1'$
$m_1' : w_3$	$w_1' : d_3$	$d_1' : m_1$
$m_2' : w_1$	$w_2' : d_1$	$d_2' : m_2$
$m_3' : w_2$	$w_3' : d_2$	$d_3' : m_3$

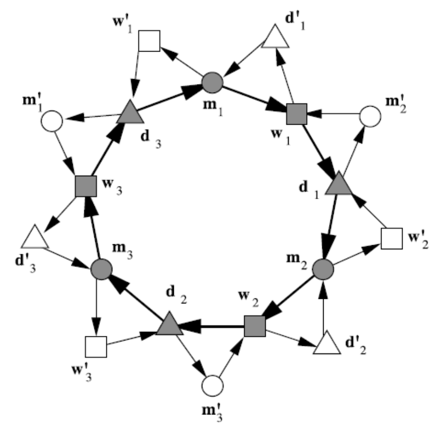


Fig. 16 Non-existence of 3DSM stable matching for n=6

The main questions that remain unsolved are

a. whether there exists an instance of cyclic 3DSM that admits no stable matching, and
b. whether there is a polynomial time algorithm to find such a matching or report that none exists, given an instance of cyclic 3DSM.

3-WAY KIDNEY EXCHANGE PROBLEM This is an application of the Circular Stable Matching. Circular Kidney Exchange problem has been first introduced by Knuth with his question “It is not always the case that we have 2 pools of datasets, so can there be any way that Stable Matching ”In Circular Stable matching instead of 2 pools of data we have 3/ more pools of datasets. Taking an example of a 3 pool dataset say we have men preferring women, women preferring dogs and dogs preferring men, where each entity from each dataset prefers another entity from other dataset and this way entity dataset-preferred dataset form a circular representation. The goal is to organize them into family units (man, woman, dog) so that no three of them have incentive to leave their assigned family members to join in a new family. The family units are in the form of a triplet. Supposing that we have family units as $\{(m1,w1,d1), (m2,w2,d2), (m3,w3,d3)\}$ in the matching set M, if $m1$ prefers $w2$ to $w1$, $w2$ prefers $d3$ to $d2$, and $d3$ prefers $m1$ to $m3$, then $(m1, w2, d3)$ becomes a blocking triple. But it may also be the case that $w2$ prefers $d1$ to $d2$. Then $(m1, w2, d1)$ can also be conceived as a (weaker) blocking triple, since only $m1$ and $w2$ are really preferring each other in such a triple, while $d1$ is indifferent. Hence, the concept of blocking pairs is more complex in circular stable matching than in basic stable matching.

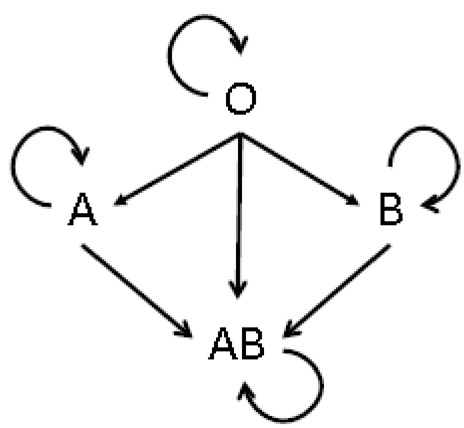


Fig. 17: Blood Compatibility

This concept has been extended to be applied in the field of medical science for Kidney Exchanges. Currently, 118, 617 patients are waiting for a prospective donor for vital organ transplantation, and among these about 93, 000 are waiting for Kidney Transplantation. In most of the cases it is the family member who is interested to donate a kidney. But, unfortunately in many cases, they fail to match the tissue type or blood group type of the patient. Such cases are named as Tissue Incompatibility and Blood Group type Incompatibility respectively.

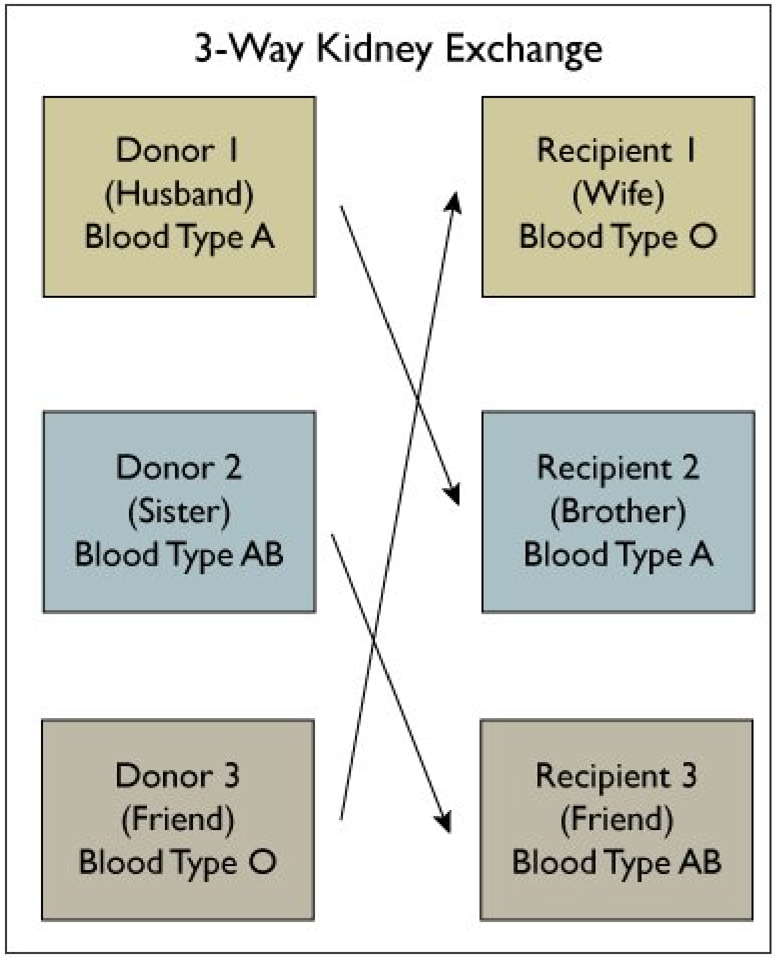


Fig. 18: 3-WAY KIDNEY EXCHANGE PROBLEM

Such a (patient donor) pair is referred to as Incompatible Pair. Figure 5 represents the blood compatibility issues. To this Circular Stable Matching comes forward with solution of 3-Way Kidney Exchange. The Figure 6 below shows an example of how 3-Way Kidney Exchange works. Here, we have 3 donors and three recipients. Recipient 1 (wife) having the blood group O seeks a kidney here. Both husband and husband's sister try giving the kidney, but both fail due to blood group incompatibility and person with Blood Group O can only receive from another person with the same blood group, thought he/she can give to anybody. Recipient 2 (wife's brother) having blood group A receive from the husband and Recipient 3 (Wife's brother's friend) having the universal receptor type blood group can receive from anybody (Donor 2).Stable matching concept is applied in 3-WAY KIDNEY TRANSPLANT, as matching is composed of oriented triples. Here, we write such a triplet as ($k1$,$k2$,$k3$) to denote that $k2$,$k3$,$k1$ are the successors of $k1$,$k2$,$k3$, respectively. Here, each triplet (k1, k2, k3) represents a couple which is often a married couple in case on stable matching, but here it consists of a person needing a new kidney (patient) and a potential kidney donor (donor). If $k2$ follows $k1$ in a triplet, then the donor from the (patient, donor) pair $k2$ will be donating a kidney to the recipient of $k1$. Thus, it is $k1$'s preference (degree of compatibility) that needs to be considered. At this point, a point to note here is that an oriented couple matching set ($k1$,$k2$,$k3$) can be a blocking triplet itself ($k1$,$k3$,$k2$), if $k1$ prefers $k3$ to $k2$, $k3$ prefers $k2$ to $k1$, and $k2$ prefers $k1$ to $k3$. Such phenomenon has been an open stable matching problem for the researchers in the stable matching literature till now. For tissue incompatibilities, the serum from a patient and lymphocytes from a donor must be physically mixed to affirmatively determine compatibility issues before a transplant can take place. Such a test is known as the Human Leukocyte Antigen (HLA) Test. Each such test requires a non-trivial amount of blood; therefore it is not feasible to exhaustively determine all compatibilities in an exchange with hundreds of participants. But, here a problem lies in coordinating the subsets of the test domain as the incompatibility pairs led from blood type incompatibilities and tissue incompatibilities may be located at multiple hospitals across India. For this till now there is centralized national testing centre. Therefore, instead of real data compatibility tests, virtual compatibility tests based on computer programming by running the Circular Stable Matching algorithm is done, which is then later verified by real compatibility tests before transplantation.

Further, 3-Way Kidney Exchange has been extended to the case where there are multiple donors for a single patient interested to donate. For example, if we have a child (patient) then both the parents may be interested to donate. In this case we have a problem similar to ties in SMP and it is solved in the same way too by arbitrarily selecting one. When represented in the form of graphs, a back arc always implies an embedded pair-wise exchange.

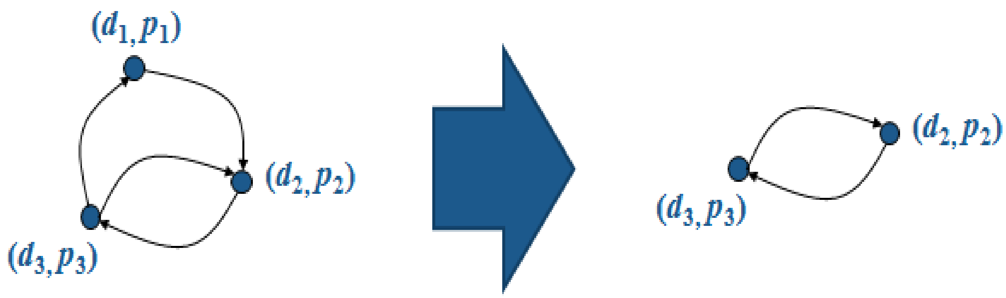


Fig. 19: Embedded Pair-wise Exchange

In the above figure, if (d1, p1) drops, then the back arc provided by (d2, p2) to (d3, p3) could still carry on the process. It could be extended with (d1, p1) as a middle node, but with a little risk associated, as longer chains are more risky than shorter ones. After studying the basic properties of Kidney Exchange Problem, researches set some goals to achieve:

a. Maximizing Pair-wise Exchanges
b. Maximizing overall Number of transplants
c. Minimizing Number of 3-Way Kidney Exchanges
d. Maximizing Number of back-arcs.
e. Maximizing Over weight

All these problems have been solved in one way or the other by using Integer Linear Programming (ILP), by defining Maximization or Minimization Problem. Programmers have implemented these using C++, using packages including COIN-CBC (ILP solver), LEMON (maximum matching library for maximum matching), Ruby on Rails framework for web service, Google Test (Testing Framework) taking as input in X

ML/JSON format called via SOAP/RESET protocols. Output is also given in the same format. This software can be deployed on Windows, Linux and Solaris. Demonstration of this application run on the C++ platform is given in the http://kidney.optimalmatching.com.

STABLE MATCHING IN NETWORKS WITH BOOLEAN CIRCUITS Mayr and Subramanian (1989) [47]: impose fan out restrictions at each gate in the network to form a restricted network, and special cases of SMP & SRP are expressed. Proposed a generalized model by allowing the gates in the circuit to produce several output values. Multiple copies of the same output can be prohibited by setting each gate in the set to be adjacency preserving, i.e. change in the value of one input affects only one output value. The problem of deciding whether such a given network has stable configuration is NP Complete without adjacency constraint imposed. Imposing the adjacency preserving constraint stable configuration can be found in $O(n^3)$ time. In 1989, Tomas Feder [48] proposed a new fixed point approach for stable networks and stable marriages, which aimed at finding a stable configuration for a given network of boolean gates-general networks which is basically computationally hard. He studied sequential and parallel complexity of the above restricted network. Considering *m* as number of edges in the network and *n* as width of 2-SAT problem, findings are:

1. Optimal algo for finding 2-SAT instance characterizing the set of stable roommates assignments, giving all stable pairs in O(m) time. Earlier, in 1988 Gusfield took O(nm log n) time to do the same.

2. Algorithm enumerating all stable roommate assignments in optimal O(n) time using O(m) space. Gusfield took O(m) time.

3. An optimal SMP running in $O(m^{1.5}\log m)$ time. Leather and Gusfield (1985) in $O(m^2)$.

4. Proof that optimal stable roommates problem is NP Complete.

5. An optimal construction showing that every instance of 2 SAT with n variables and m clauses characterizes the set of solutions of some small stable roommates problem with O(n) people and O(m) candidate pairs.

SELFISH BIN PACKING [49] This consists of a network of two nodes i.e. source and destination connected by a set of parallel paths, each having same bandwidth capacity and a set of users. The motive of users is to route a certain amount of packets from the source to destination node following any of the links. While doing so, the user suffers a link delay equals to the total amount of total amount of flow routed through the link, hence more the flow of packets in a specific link more in the delay incurred. The cost is calculated as the fraction of the used bin space. For such a reason the users act selfishly and set their preference list for the links such that they grab the least loaded link to transfer the data and equilibrium. A pure Nash Equilibrium (NE) is a packing where no agent can obtain a smaller cost by unilaterally moving his item to a different bin, while other items remain in their original positions. A Strong Nash Equilibrium (SNE) is a packing where there exists no subset of agents, all agents in which can profit from jointly moving their items to different bins. We say that all agents in a subset profit from moving their items to different bins if all of them have a strictly smaller cost as a result of moving, while the other items remain in their positions.

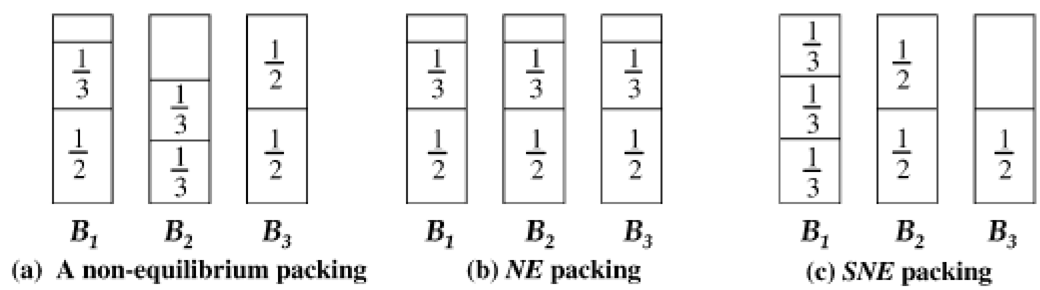


Fig. 20: (a) A packing that is not an equilibrium, as the item of size 1/3 on B1 will reduce its cost by migrating to B2. (b) A packing that is Nash equilibrium but not a Strong Nash Equilibrium, since the five items of sizes {1/2, 1/2, 1/3, 1/3, 1/3} will reduce their by deviating as shown in (c). which is a strong Nash equilibrium.

MATCHING OUTPUT QUEUING WITH A MULTIPLE INPUT/OUTPUT QUEUED SWITCH Hyoung-I1 Lee and Seung-Woo Seo [50] proposed an efficient switch architecture called multiple input/output-queued (MIOQ) switch and showed that MIOQ switch can match the performance of an output queued switch exactly with no speed up of any component. They have proposed a stable strategic alliance (SSA) algorithm that can produce a stable many-to-many assignment, and proved its finite and deterministic properties. Further SSA has been applied to the scheduling of a parallel MIOQ switch with two parallel switches to show the stability. A simple algorithm requiring at most 2N steps is designed to avoid conflicts in a parallel switch, such that each input output pair matched by the SSA algorithm must be mapped to one of the two crossbar switches as shown in Fig. 21.

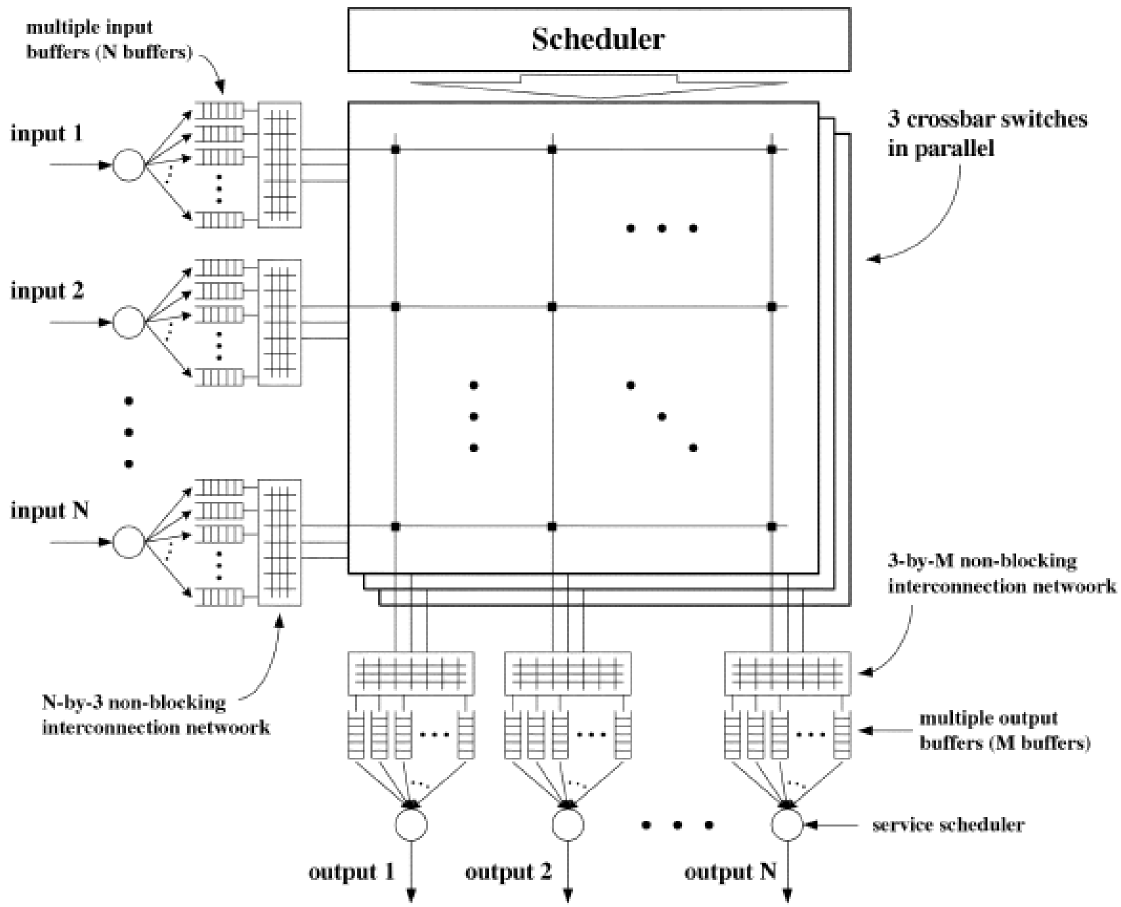


Fig. 21: A N X N PMIQ switch architecture with three crossbar switches in parallel

Redundant buffering scheme requiring two memory devices only instead of *N* physically separate memories, relieves the implementation burden of *N* input buffers being accessed simultaneously. Fig. 22 shows a snapshot of *2 X 2 PMIOQ* switch with two crossbar switches at the arrival of a cell denoted as *A-4*, where *A* denotes its output port and 4 represents it's time to leave. Since its output *A* has 2 cells out of which time to leave values are lower than 4, its output cushion is 2 which is located on the third position of the input priority list of *X*. According to its TTL value, its location in the output priority list of A is determined. The preference lists for scheduling are also given here as complete ordered list of head-of-line cells.

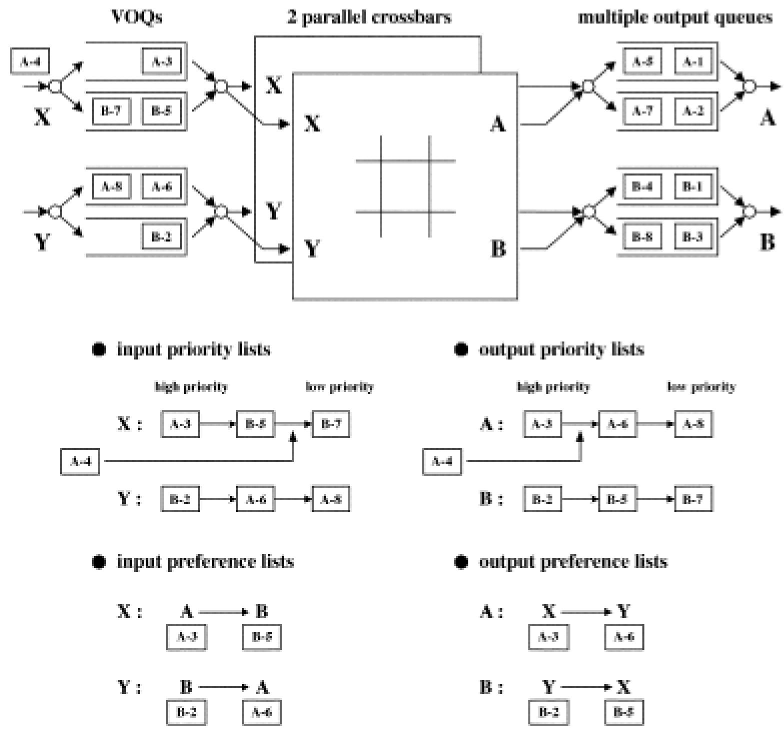


Fig. 22 Example of the operation of a 2_2 PMIOQ switch with two crossbar switches for output queueing emulation in a time slot.

On the output side, the PMIOQ switch keeps track of time-to-leave (TTL) for each head-of-line cell of output buffers, and sends out cells at their time to leave during the departure phase. As cells of an output buffer are located in a sorted manner according to their time-to-leave values, consideration of the head-of-line cells of output buffers is sufficient for output queuing computation. Therefore, if each cell can be transferred to the output side before its time to leave, the PMIOQ switch can mimic output queuing correctly.

INTERVAL SCHEDULING [51] Supposing that a resource can be used by atmost one person at a time and we have many people queued to access it, a scheduler is needed to schedule the requests and act accordingly. Formally, we have *n* requests labeled *1, . . ., n*, with each request *i* specifying a start time s_i and a finish time f_i, with $s_i < f_i$ for all values of *i*. In this case, *i* and *j* are said to be compatible if the request intervals do not overlap; i.e. either request *i* is for a earlier time interval than request *j* ($f_i < s_j$), or request *i* is for a later time than request j ($f_j < s_i$). We can say that a subset A of requests is compatible if all pairs of requests $i, j \in A$, $i != j$ are compatible, which is similar to stability without any blocking pairs. Therefore the aim is to select a compatible subset of requests of maximum possible size. An instance of Interval Scheduling Problem is shown in Fig. 23.

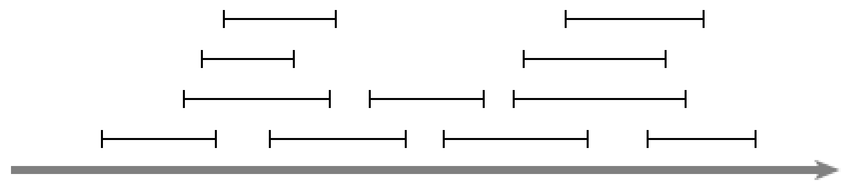


Fig. 23 An example of Interval Scheduling Problem

This problem can be solved by general implementation of GSABASIC, with the compatible requests ordered in a heuristic way and then greedily processing them in one pass, selecting as large as compatible subset as it can, providing an optimal solution.

INDEPENDENT SET [52] Given a graph G (V, E), we say that a set of nodes S ⊂ V is independent if no two nodes in S are joined by an edge. Therefore, here our aim is to find a independent set that is as large as possible. This problem encodes any situation in which we are trying to choose from among a collection of objects and there are pairwise conflicts among some of the objects.

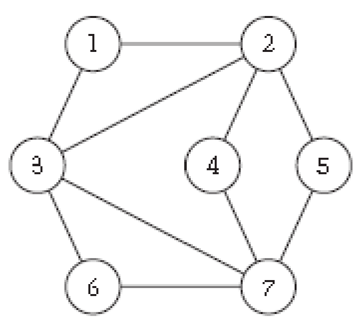
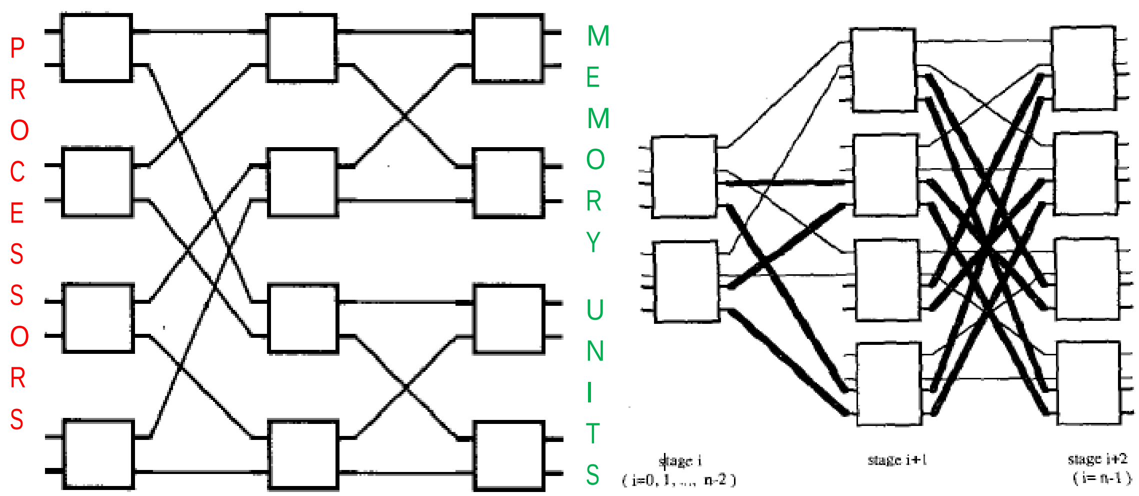


Fig. 24 A graph whose largest independent set is of size 4

Both Interval Scheduling and Bipartite Matching can be encoded as a special case of Independent Set Problem.

PROCESSOR-MEMORY MATCHING INTERCONNECTION NETWORKS [53] Based on different criteria the processors can prioritize the memory units to fetch data shown in Fig 25.



(a) (b)

Fig. 25 (a) Three stage BIN (b) A three stage ABIN (stably matched)

OPTIMIZING HZTN NETWORK: [54] Networking optimization problem can also be solved as matching problem such as assigning channels to the users, flows in wireless scheduling, in video-streaming systems where maps the video segments to servers. Perhaps due to more technology evolution networks has become complex. However its effectiveness depends upon the availability of complete and accurate information which in practice may not be possible such as parameter serving as input may be perverted due to delay, noise or inaccurate measurement which may lead to drift in computed optimal solution from the actual one. Hence stable matching is used to tackle the problem where preference list poses as each element interest and stability provide the solution. Ernst W Mayer studied the complexity of circuit problem and network stability conditioned when fan-out is restrained. Firstly problem was studied in terms of circuit value where job is described as Boolean circuit and is given input so that can calculate the value of its output. Further the problem was defined as network stability where network is Boolean circuit with feedback. Hence network stability is defined as given a circuit and its input the job is to assure whether there is a consistent way to assign values to the links of network. Further they researched on the complexity of network stability and circuit value depending on the properties of gate consisting in the network. It has earlier been shown the stable matching problem as stable consequences of –networks. In new approach, Ashok Subramanian stated the consequence where he included easy proof to old theorem and further defined some algorithm clearly differentiating between stable marriage problem and stable roommate problem. Also proved NP- complexity of three sided stable marriage, CC-problem of several other stable problem and provided parallel algorithm reducing the stable marriage problem to assignment problem.

Nitin and Ashok S proposed the relationship between Stable Matching and MIN stable problem. They idea behind the concept was the proof of network stability problem. In general, they stated that network stability problem is NP-complete problem but if network is a multi stage interconnection network then network stability problem is equivalent to stable matching problem. They considered different classes of MIN such as irregular regular hybrid ZETA network (HZTN), Quad tree network and regular augmented shuffle exchange network, etc as example and prove the stability using stable matching approach.

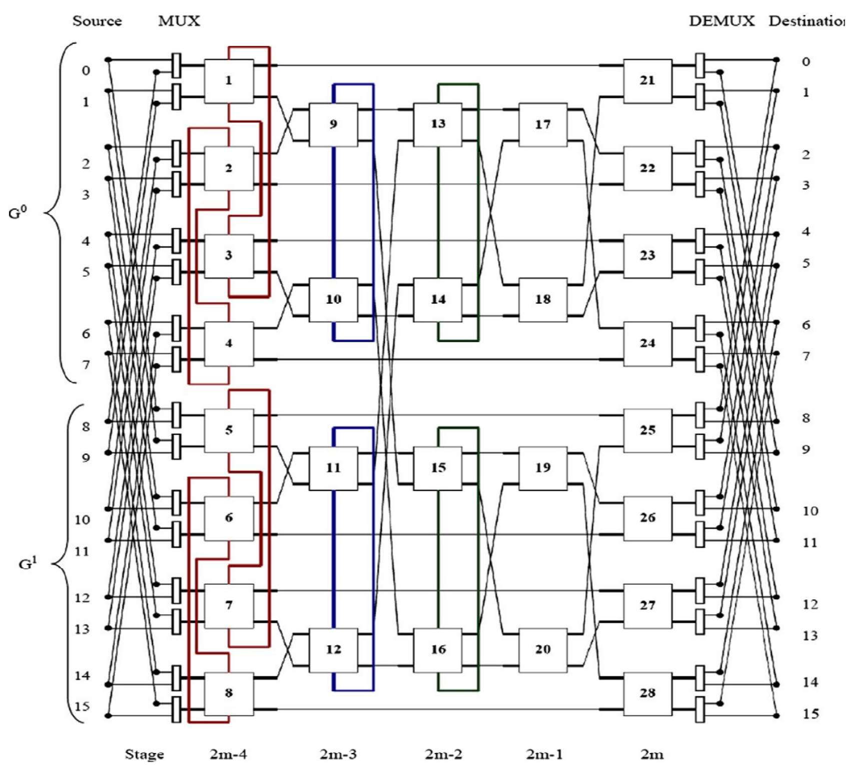


Fig. 26 A 16 X 16 Hybrid ZTN

These two scientists proved the stability of MIN by providing two algorithms- (1) The first algorithm generates the minimum preference list in $O(n^2)$ time shown in Fig. 27. (2) Another algorithm produces the set of stable pairs of switching elements derived from minimum preference list in O(n) time. Further they also presented the issues of ties between the optimal pairs.

GEN_PREF_LIST

INPUT Based on shortest path provide preference list of switching element

OUTPUT Generation of priority preference list.

PRECONDITION Only those nodes are considered for the list which they are connected to.

POST CONDITION Optimized preference list should be brought forth.

1. Stable_Pair = True;
2. **For** each switching element say node *i*
3. **For** each switching element say node *j*
4. **If** (node *i* prefers node *j* and node *j* prefers node *i* and both are connected with each other through shortest path)
5. **Then** node *i* and node *j* mutually exist in the list.
6. **ElseIf** (node *i* and node *j* have tie)
7. **Then** order their list element
8. **Else** (node *i* and node *j* do not have shortest path among each other)
9. Print "node *i* and node *j* do not have stable pair"
10. Stable_pair = false
11. **End If**
12. **End If**
13. **End for**
14. **End For**
15. Print "the generated preference list"
16. **EXIT**

STABLE_PAIR_SELECTION

INPUT: Switching Element Preference list.

OUTPUT: Stable pair

PRECONDITION: Each element provide list containing the connection with all other element.

POSTCONDITION: Stable Pair is produced for each switching element.

1. **For** each switch element node SE
2. Engaged(SE) = false
3. **End for**
4. **While** (SE not Engaged)
5. **For** each Switch Element SE *J*
6. If Switch Element SE *J* is not yet engaged
7. Then SE *I* = highest on SE J list and not engaged
8. Stable _ pair = ADD(SE *J*, SE *I*)
9. **End If**
10. **End For**
11. **End While**
12. Print "Stable Pair List"

Fig. 27 Algorithms for generating Preference Lists and making a Stable Match

HZTN consist of 2^n sources and 2^n destination where $n = \log_2 N$ and $m = \log_2(N/4)$. Network is constructed with identical group of switching element say G^N where N is either *0* or *1*. Each group is organized based on the most significant bit of the node-node terminal. Thus most significant bit of *0* comes under G^0 group and other falls into G^1 group. Each node is connected to both the group using multiplexer and de-multiplexer. To apply stable matching approach first we need to derive the preference list thus uses the path-length algorithm [55] for each source-destination pair. The preference list hence generated is shown is Fig 28.

SE 1	21	8	3	23	13	17	22	24	10	14	18	15	19	26	28	16	20	27	25
SE 2	22	9	4	24	13	17	22	24	10	14	18	15	19	26	28	16	20	27	25
SE 3	23	10	1	21	14	18	9	13	17	22	24	16	20	25	27	15	19	26	28
SE 4	24	10	2	22	14	18	9	13	17	22	24	16	20	25	27	15	19	26	28
SE 5	25	11	7	27	15	19	26	28	12	16	20								
SE 6	26	11	8	28	15	19	12	16	20	25	27								
SE 7	27	12	5	25	16	20	11	15	19	26	28								
SE 8	28	12	6	26	16	20	25	27	11	15	19								
SE 9	13	17	22	24	10	14	18	21	23	15	19	26	28	16	20	25	27		
SE 10	14	18	21	23	9	13	17	22	24	16	20	25	27	19	26	28			
SE 11	15	19	26	28	12	16	20	25	27										
SE 12	16	20	25	27	15	19	26	28											
SE 13	17	22	24	14	18	21	23												
SE 14	18	21	23	13	17	22	24												
SE 15	19	26	28	16	20	25	27												
SE 16	20	25	27	15	19	26	28												
SE 17	22	24																	
SE 18	21	23																	
SE 19	26	28																	
SE 20	25	27																	

(a)

(1, 21), (2, 22), (3, 23), (4, 24), (5, 25), (6, 26), (7, 27), (8, 28), (9, 13),
(10, 14), (11, 15), (12, 16), (13, 17), (14, 18), (15, 19), and (16, 20)

(b)

Fig. 28 (a) Generated Preference List and (b) Optimal Pair

By using STABLE_PAIR_SELECTION algorithm shown is Fig. 27 following optimal pair are generated as shown in Fig 6 shortlisted from HZTN preference list in Fig 28. The possible route path are shown in Table1 considering route path from source *0* to destination *0,* depicting all possible routes and all possible path length.

TABLE 1 ROUTING TABLE WITH PATH LENGTH CALCULATIONS

ROUTES	PATH -LENGTH
SE1 – SE21	2
SE1-SE9-SE13-SE18-SE21	5
SE1-SE3-SE10-SE10-SE14-SE18-SE21	5

Table 2 shows the comparison between different multi stage interconnection networks using the algorithms in Fig. 27 indicating that regular MINs are more stable than irregular.

TABLE 2 COMPARISON OF DIFFERENT MINS BY APPLYING STABLE_PAIR_SELECTION AND GEN_PREF_LIST

MINS	NO. OF TIES	TOTAL NUMBER OF SWITCHING ELEMENTS	MAXIMUM PATH_LENGTH	NEGLECTED PAIRS	MIN STATUS (STABILITY)
HZTN	4	16/28	5	4	Low
QTN	6	16/26	5	2	Intermediate
ASEN	4	16/24	3	0	High
ABN	3	8/16	2	0	High
CLN	4	16/24	3	0	High
GMIN	0	20/28	3	0	High
3DGMIN	3	20/28	3	2	Intermediate
3DCGMIN	0	24/32	3	0	High

3. MOTIVATION

Stable matching is perhaps one of the most important functions in many areas, e.g., economic markets. The stable marriage problem, introduced by Gale and Shapley, is best known as one of the most interesting and successful abstractions. In the stable marriage problem, a good marriage does not induce any unstable pairs. When the problem is extended beyond a one-to-one setting, e.g., the college admissions problem, it is more generally referred to as the stable matching. Two-sided matching provides a natural model in both social and economic areas. However, in some cases, the two-sided matching fails to model the real situation. So three-sided matching is needed for many problems, e.g., the supplier–firm–buyer model for market, where the value is created by the matching of a supplier, a firm and a buyer. Another example is the kidney exchange problem, where the blood type of both patients and donors can be A, B or O, and need to be compatible. Matching problems also exist pervasively in computer networking area, ranging from assigning channels to users and flows in wireless network scheduling, to mapping video segments to servers in video-on-demand streaming systems. Many networking applications or services also show the operations of assignments among three-sided agents. For example, in a sensing service system, based on requests of users, sinks/servers will collect information from appropriate sensors, performing aggregation or other operations, and forwarding the results to users. In our work, we will advocate the use of three-sided stable matching as a general framework to facilitate the networking for three-sided networks. The three-sided networks are referred to as the systems providing access to data sources for users, e.g., sensor networks for environment monitoring, video streaming networks for surveillance. They usually involve three different kinds of agents, i.e., the sources for generating service data, the middle servers for adaption/optimization/storage services and the users who request those service data. We model the Three-sided Matching with Size and Cyclic preference (TMSC) problem for such three-sided networks in a parallel manner preserving the merit of stable matching. In TMSC, preferences are used to model each agent’s interest, and stability serves as the solution concept.

4. BASIC CONCEPTS AND NOTATIONS

Basically, the stable matching problem considers two sets *M* (set of men) and *W* (set of women) each of size *n*. M is a matrix of n-men along with their respective preference list for women. Similarly, W is a matrix of n-women along with the respective preference list for men. Here, we are considering the preference lists to be complete and strictly ordered. A complete list is such that a man needs to specify the ranks for all of his partners that are participating in the game, and a strict ordering of lists puts a bound that man needs to be clear about his thoughts for the preferences of his partners and therefore he cannot assign the same rank to more than one partner. In any instance of the matchmaking problem we uniquely match each man in set M with its woman (partner) in the set W for a man-oriented approach and vice versa if it is a woman-oriented approach, which means that GSA is partial. Either it favors men leading to a man optimal and woman pessimal solution or the other way round. To achieve global optimality with respect to both the sides we have *egalitarian approach* with a time complexity, $O(n^4)$. Given a problem-instance, *Ii* a matching □ is a pairing of man Mi to woman Wj. If (m_1, w_3) belongs to □, then we can say that m_1 and w_3 is a couple. A complete 1:1 matching of each man in set M to each woman in set W uniquely is known as a *marriage*. If a man and a woman in different couple in the matching set □ prefers to each other to their present partner then we say we have a *blocking pair* and the marriage is not stable. Therefore, a *stable matching* is a marriage with no blocking pairs. *Rank* refers to the priority (position) in a person's preference list of his or her partner. We will denote the rank of woman w_j for man m_i in his preference list as $Rm_i(w_j)$ and score as the sum of all the ranks in □, denoted by $S_{□}(I)$. In our approach we will be making match in an optimal, conflict free and stable way between 3 agents i.e. users (U), sensors (D) and servers (S), each preferring the other in a cyclic way as shown in Fig. 30. U is a set of users along with their preference list for sensors. Users are a group of entities seeking services from the server. But as a client can never contact the server without any middle interface, they need to establish connection with the sensors first. Users specify sensors in their preference list, preferring the one with maximal strength. Signal strength is a cumulative factor, which is a combination of mostly minimal distance and the maximal available bandwidth by any sensor.

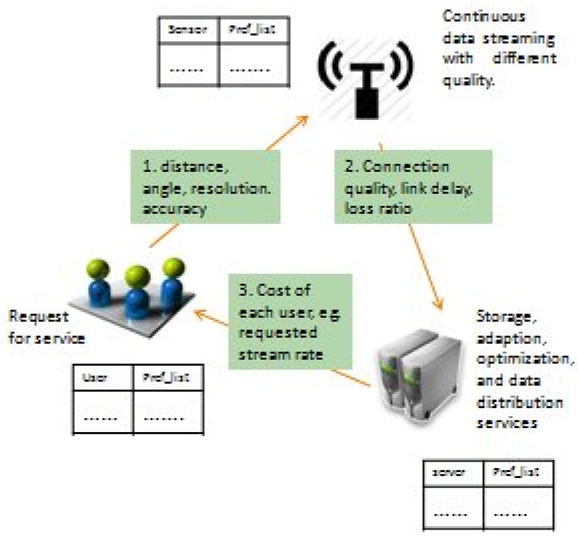


Fig. 29 Three-sided cyclic wireless sensor network with cyclic preferences

Distance here is measured in terms of Euclidean Distance, which is given by the formula:

$$d_E\,(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

$$= \sqrt{\textstyle\sum_{i=0}^{n}(a_i - b_i)^2} \qquad \textbf{(1)}$$

The less is the distance of the sensor from the user; more will the resolution and accuracy expected; unless there are already many users using the available bandwidth provided by the sensor. Therefore, we take strength as the prioritizing parameter. Signal strength has 5 levels: very poor, poor, good, very good and excellent represented by 5 vertical lines.

D is a set of sensors along with their preference list for servers. Sensors continuously stream the data received with different quality requested by the user. Here the parameter to order the servers in the preference list is taken to be connection quality, which is determined by the considering the propagation time. Less is the propagation time, better will be the quality of the connection. Propagation time (P_t) is given by:

Propagation time = *Frame Serializability Delay* + *Link Media Delay* + *Queuing Delay* + *Node Processing Delay* (2) where;

Frame Serialization Time = *Packet size (bits)/ Link Data Rate (bps)* (3)

Frame Serialization Delay in the time required to put the data on the hardware (physical wire), and is a constant based on the access rate of interface. This value is set by the manufacturer and cannot be altered; therefore the data frame can pass at only serialization rate.

Link Media Delay = *Link Distance (meters)/ Medium Propagation Speed* **(4)**

Manufactures generally select the link medium either to be copper or fiber. Speed for a copper wire, is 210 x 10^6 mps and that of fiber is 300 x 10^6 mps.

Queuing Delay = Queue Depth (bits)/ Link Data Rate (bps) **(5)**

We are assuming that the link is not congested; hence there is no queue depth and the factor, Queue Delay reduces to zero. Propagation delay (P_d) is different for each node and is determined specifically.

S is a set of servers along with their preference list for users and capacity of each to accept a definite number of user's stream requests. The capacity of a server is determined by processing ability and the bandwidth of the server. Based upon the cost of the data requested by the users, servers order their preference list. The cost of user's request is measured in terms of stream rate of the data requested. Server is more inclined to serve

TABLE 3 RECOMMENDED BIT RATES FOR LIVE STREAMING OF HIGH AND LOW MOTION VIDEO EVENTS

VIDEO SIZE TYPE	ASPECT SIZE 4:3	ASPECT SIZE 16:9	V_RATE/A_RATE	BIT RATE (KBPS)
QCIF (176 x 144)	144 x 108	192 x 108	32/16	48
	192 x 144	256 x 144	80/16	96
CIF (352 x 288)	288 x 216	384 x 216	268/32	300
	320 x 240	384 x 216	468/ 32	500
D1 (720 x 486)	640 x 480	852 x 480	768/ 32	800
	640 x 480	852 x 480	1168/ 32	1200
HD (1280 x 720)	-	1280 x 720	1768/ 64	1800
	-	1280 x 720	2336/ 64	2400

users with a request for lower bit-rate than with a higher value optimally. Table 3 lists the required stream rate for a high motion event like movies, sports, news etc. Based upon the connection type a user is needed to input the required stream rate. Among each set of agents i.e (U, D), (D, S) and (S, U) we aim to apply parallel GSA ($GSA_{PARALLEL}$), so as to match with a lower time complexity of *s*. But, $GSA_{PARALLEL}$ fail to produce proper results in worst-case instances. In the basic Gale Shapely algorithm it has been clearly specified that, in a man-oriented stable matching, it should always be man-optimal and this adds to the merit of GSA_{BASIC} also. But in some instances, we have women getting better partners than men leading to a man-pessimal matching and a woman optimal match. For such instances merit of the Gale Shapley algorithm cannot be used efficiently. Here we illustrate the scenario with the help of an example. Suppose we have set of 4 men and 4 women. Each preference list is ordered (ranked) in increasing order from left to right as shown in table 4. Lower the priority higher is the preference.

TABLE 4 PROBLEM INSTANCE FOR WORST CASE IN GSA_{BASIC}

m_i	Pref. List (PLm_i)				w_j	Pref. List (PLw_j)			
m_1	w_1	w_2	w_3	w_4	w_1	m_2	m_1	m_3	m_4
m_2	w_3	w_1	w_2	w_4	w_2	m_1	m_2	m_3	m_4
m_3	w_2	w_4	w_3	w_1	w_3	m_4	m_3	m_1	m_2
m_4	w_2	w_3	w_4	w_1	w_4	m_3	m_2	m_1	m_4

GSA_{BASIC} ends us with the matching set, □ = {(m_1, w_2), (m_2, w_1), (m_3, w_4), (m_4, w_3)}. As we can see in the men-table (Table 5) we have all the men getting paired up with their second preferences, and the women get the men from their first preferences (Table 6).

TABLE 5 MEN TABLE

m_i	Pref. List (PLm_i)			
m_1	w_1	w_2	w_3	w_4
m_2	w_3	w_1	w_2	w_4
m_3	w_2	w_4	w_3	w_1
m_4	w_2	w_3	w_4	w_1

TABLE 6 WOMEN TABLE

w_j	Pref. List (PLw_j)			
w_1	m_2	m_1	m_3	m_4
w_2	m_1	m_2	m_3	m_4
w_3	m_4	m_3	m_1	m_2
w_4	m_3	m_2	m_1	m_4

Gale Shapely Algorithm says that in case of a man-oriented approach a man always gets its best possible partner and a woman its worst possible partner i.e. it should be man-optimal and woman-pessimal. But the result we got in this case is a man-pessimal and woman-optimal solution. As the result shows, we have women happier, with a score of 4 than men when men (score is 8) initiate the proposal, contrary to what GSA says. This should not happen unless any woman cheats by changing her preference list after anticipating the men' s order of proposals and choosing the cheating strategy for herself to get the man she desires. But in this case the women gain a heavier side of the balance as the men (proposing entity) cannot do anything to save them from deception; therefore proposed the cheating by men approach. The authors have tried eradicating the worst case scenario for Gale Shapely Algorithm by allowing small changes in the preference lists. At first, only one man's preference list was allowed to change. They have considered two variants of the problem:

1. *Optimization Variant*: Can we find a man and the way in which he needs to change his preference list that can cause maximum improvement.
2. *Decision Variant*: Can we find a positive improvement?

Both these variants put a restriction that no man should be assigned a worst partner than the man optimal stable matching, i.e. in a man oriented approach; it can never derive a man pessimal result. The time complexity for the optimization variant was given to be $O(n^3)$ and that of the decision variant is given to be $O(n^2)$. To derive the decision variant the structural property of stable matching is used. Graphs are generated for the women preferences and then it is checked to see, if any cycle exists then the man-oriented approach has led to a man pessimal solution and hence needs to be taken care of, if 'no' then the algorithm ends. Extension of this problem has considered changing the preference list of k-men and the derived general formula is given by $O(n^{2k+1})$ and $O(n^{k+1})$ respectively. The proposed algorithm showed best results when k = n, i.e. we are allowed to change the preference lists of all the men. In such a case the optimization variant time complexity is found out to be and the decision variant time complexity be $O(n^2)$.

5. PROPOSED APPROACH

The above result we got is the worst case scenario represented in table 2 where the proposing party (table 3) who is expected to be happier than the non-proposing one (table 4) is sad rather. It has been proved that if there are 16 men and 16 women then the probability that the worst case occurs is 10^{-45}, which is very low. Parallel algorithm is based upon divide and conquer principle having a time complexity of n^2-2n+ [log n] as stated before, which is better than the time complexity of basic Gale-Shapely Algorithm. But, parallel algorithms do not work for the worst case. We can avoid such worst case scenario to some extent by following our proposed algorithm, Modified GSA (MOD_{GSA}).

ALGORITHM: MODIFIED GSA (MOD_{GSA})

Input: The problem instance with the men matrix M and the women matrix W along with their preference lists.

Output: A man-optimal matching set, ɱ$_i$ for a man-oriented approach.

Precondition: The problem instance should produce the worst case scenario.

1. Calculate the score for the problem instance I_0
2. **For** each pair in matching set ɱ$_0$ do
3. Delete the pair p_i
4. Form the matching set ɱ$_i$ using GSA_{BASIC}
5. Calculate the score, $S_ɱ(I_i)$
6. **End for**
7. Delete the pair p_i for which the score, $S_ɱ$ is minimum
8. Output the matching set ɱ$_i$ for which the pair has been deleted.

In this algorithm we are taking as input a matrix of M for men and W for women, and their preference lists ordered according to their priority. We apply GSA on the basic problem instance I_o and we denote the matching set found, by ɱ$_0$. For say, we have a set of 4 men and 4 women then the matching set formed will have 4 pairs with each man paired with his respective partner, we then denote the matching set ɱ$_0$=[p_1, p_2, p_3, p_4] where p_i denotes a pair i. Therefore, we can say that the matching set is a matrix with n-pairs for n being the size of the problem. Though the problem size is considered n × n, but for simplicity we will consider it n throughout the proposed solution.

The for-loop in the Modified Gale-Shapely Algorithm runs for each pair which is given by the problem size only i.e. n. For each pair we delete the pair first. Then we apply GSA to the new matrix set of (n-1) men and (n-1) women, leading to a matching set $?_i$. Finally we calculate the score of $?_{i,}$ as $S_{?i}(I_i)$. We continue to do so for the entire pairs p_i in the original GSA matching set, and select to delete the pair with the minimum score, S_{min}. Therefore, the GSA matching set retained now has the minimum score. In the matching set ?, the score is found as the sum total of all the ranks of the partners in the preference list of the proposing party.

Time Complexity: The time complexity $T(n)_{GSA}$ of the above algorithm is given by $O(n^3)$.

Proof of Complexity or Correctness: The Gale Shapely algorithm takes $O(n^2)$ for a problem size n. We have the for-loop run for each pair. For a problem size n we always end up having n pairs. Therefore, we get the complexity to be calculated as:

Time Complexity, $T(n)_{GSA} = (n-1)^2 \times n$

$$= O(n^3)$$

We can improve this time complexity by following parallel GSA ($MOD_{P\text{-}GSA}$) instead of basic GSA at line 4. At line 1 we have calculated the score i.e. sum of the ranks of the matched pair. Here we have made possible for the parallel algorithm to execute successfully with high probability in case of a worst case scenario by deleting or ignoring one couple from the matching set $?_0$.

Here we will consider the table 1 and based upon it we will describe our algorithm for MOD_{GSA} in a stepwise manner. We will consider the Modified Parallel GSA denoted by $MOD_{P\text{-}GSA}$ in the next section of implementation.

For problem instance in table 3 the following steps describe the flow of the algorithm.

STEP 1: Applying GSA we get the matching $?_0 = \{(m_1, w_2), (m_2, w_1), (m_3, w_4), (m_4, w_3)\}$ having the score, $S_?(I_o)$= 2+2+2+2=8.

The score we will calculate at a later stage should come less than this as we are trying to maximize happiness for men. This constraint would verify the correctness of our algorithm as less is the score more is the happiness.

STEP 2: For deletion, we need to consider each pair in M. As for a problem size n we always end up having n-pairs, this for loop will run for n-times.

STEP 3: We proceed first by considering $\square_0[0]$ i.e. (m_1, w_2). Now we are left with the matrix shown in table 7:

TABLE 7 REDUCED TABLE

Man_i	Pref. List (PL_{mi})	$Woman_j$	Pref. List (PL_{wj})
m_2	w_3, w_1, w_4	w_1	m_2, m_3, m_4
m_3	w_4, w_3, w_1	w_3	m_4, m_3, m_2
m_4	w_3, w_4, w_1	w_4	m_3, m_2, m_4

STEP 4: Applying GSA, $\square_1 = \{(m_2, w_1), (m_3, w_4), (m_4, w_3)\}$

STEP 5: For the above reduced problem instance I_1, score is given by, $S_{\square}(I_1)= 5$

STEP 6: Similarly doing it for all other pairs in $\square$, we have

Delete (m_2, w_1): $\square_1=\{(m_1, w_2), (m_3, w_4), (m_4, w_3)\}$, $S_{\square}(I_2)= 6$

Delete (m_3, w_4): $\square_1 =\{(m_1, w_1), (m_2, w_3), (m_4, w_2)\}$, $S_{\square}(I_3)= 3$

Delete (m_4, w_3): $\square_1 = \{(m_1, w_1), (m_2, w_2), (m_3, w_4)\}$, $S_{\square}(I_4)= 4$

STEP 7: Choosing the pair with minimal score we delete (M_3, W_4), and we are left with the matrix, shown in table 8:

TABLE 8 RESULT TABLE

Man_i	Pref. List (PL_{mi})	$Woman_j$	Pref. List (PL_{wj})

m_1	w_1, w_2, w_3	w_1	m_2, m_1, m_4
m_2	w_3, w_1, w_2	w_2	m_1, m_2, m_4
m_4	w_2, w_3, w_1	w_3	m_4, m_1, m_2

The table 8 clearly shows that now the men get their first preferences and women either their second or third, resulting into a man-optimal and therefore woman pessimal solution. As we can see here, our problem size has been reduced to (n-1), but as we go on increasing the value of n, this hardly matters, if by doing so we get an overall happiness and preserve the basic property of Gale-Shapely Algorithm by reducing the occurrence of a worst case. Parallel GSA follows divide and conquer principle to solve the matchmaking problem in a parallel way taking (n^2-2n+$\log_2$n) steps, where n is the size of the main problem.

ALGORITHM: MODIFIED GSA ($MOD_{P\text{-}GSA}$)

Input: The problem instance with the men matrix M and the women matrix W along with their preference lists.

Output: A man-optimal matching set, m_i for a man-oriented approach.

Precondition: The problem instance should produce the worst case scenario.

1. Calculate the score for the problem instance I_0
2. **For** each pair in matching set m_0 do
3. Delete the pair p_i
4. Form the matching set m_i using $GSA_{PARALLEL}$
5. Calculate the score, $S_m(I_i)$
6. **End for**
7. Delete the pair p_i for which the score, S_m is minimum
8. Output the matching set m_i for which the pair has been deleted.

As the name inc ... ering (merging) phase. The division of the problem into sub-problems led to a tree like structure and problems at the same tree level are solved in a parallel fashion to produce a partial matching set which is then merged to form a higher level match. The conflict where a single man is matched twice at the same level with two women is solved by consulting the women's preference list. This whole process continues until we get the final result.

Parallel GSA does not work is a worst case scenario, therefore the input to $MOD_{P\text{-}GSA}$ i.e. the matching set ɱ$_0$ is calculated following the GSA_{BASIC}, which takes at most n^2 number of steps in a worst case. Inside the algorithm where we obtain the matching set ɱ$_i$ we will use the parallel GSA. Even here there is a chance that the worst case scenario may occur. But it has already been stated before that the chances of the occurrence of worst case are very rare and the rarity increases even more as we are searching for a worst case within the worst case.

ALGORITHM: MODIFIED GSA ($MOD_{P\text{-}GSA}$)

Input: The problem instance with the men matrix M and the women matrix W along with their preference lists.

Output: A man-optimal matching set, ɱ$_i$ for a man-oriented approach.

Precondition: The problem instance should produce the worst case scenario.

1. Calculate the score for the problem instance I_0
2. **For** each pair in matching set ɱ$_0$ do
3. Delete the pair p_i
4. Form the matching set ɱ$_i$ using $GSA_{PARALLEL}$
5. Calculate the score, $S_{ɱ}(I_i)$
6. **End for**
7. Delete the pair p_i for which the score, $S_{ɱ}$ is minimum
8. Output the matching set ɱ$_i$ for which the pair has been deleted.

Time Complexity: The time complexity, $T(n)_{P\text{-}GSA}$ of the Modified Parallel GSA is given by $O(n^3)$.

Proof of Complexity or Correctness: The Parallel Gale Shapely algorithm takes (n^2-2n+$\log_2 n$) number of steps for a problem size n. As for-loop runs after we delete a pair, the problem size reduces to (n-1). This for-loop runs for each pair. For a problem size n we always end up having n pairs. Therefore, we get the complexity to be calculated as:

Time Complexity, $T(n)_{P\text{-}GSA} = ((n-1)^2-2(n-1)+\log_2(n-1)) \times n$

$= (n^3)$

Next we incorporate $MOD_{P\text{-}GSA}$ into a restricted model for video on demand application as: OPTIMAL_NETWORK_MATCH which works in a parallel manner to match entities from each entity set: U, D and S and outputs a stable triplet match such that users attain maximum satisfaction. The algorithm OPTIMAL_NETWORK_MATCH matches the entities in a cyclic and stable way. It takes as input all the three sets of entities i.e. U, D, S and outputs an optimal matching set . This is the main algorithm which calls two other procedures during the course of its execution. The whole problem, *P(U, D, S)* is divided into three problem pairs: *P(U, D), P(D, S)* and *P(S, U*) and each is solved in a parallel way to generate corresponding solutions. If the solution does not abide by the *Satisfiability Rule*, then the solution along with its corresponding problem set is sent as parameter to UTMOST_SATISFACTION procedure. Finally the solution of each pair sent is merged to find out the ultimate matching (U, D, S).

ALGORITHM: OPTIMAL_NETWORK_MATCH(P)

Input: A problem P (U, D, S) where U, D and S are two users, sensors and servers entity-preference matrices.

Output: Matching Set, M (U, D, S)

Begin
Assign $S_{U,D}$ = PARALLEL_GSA ($P_{U,D}$)
If (Sat ($S_{U,D}$ (U)) < Sat($S_{U,D}$ (D)))
$S_{U,D}$ = UTMOST_SATISFACTION ($P_{U,D}$,

Parallel GSA takes as input a problem instance *P (A, B)* and returns a matching set, S_{final}. It begins by assigning each entity its first priority and then recursively dividing the problem set into equal halves until we are left with a problem of size 1, known as the Requesting Phase. Next, we start merging the problem sets to form temporary solution sets, S_{temp} which finally combine to provide S_{final}.

ALGORITHM: PARALLEL_GSA (P)

Input: A problem P (A, B) where A and B are two entity-preference matrices.

Output: Matching Set, S_{final}

1. **Begin**
2. **Repeat**
3. Divide problem, P into two equal sized sub-problems.
4. **Until** ($P_{size} = 1$)
5. **Repeat**

At each merging level, we see if there arises any conflict in finding S_{temp}. Conflict refers to the same entity being requested by more than one entity. In case of servers, which can accept request from more than one sensor, a conflict arises when number of requests exceeds the server's capacity threshold. At line 7 in case of any conflict, the requesting entity is added to the *Rejected Set*, 'R_s'. For each entity in R_s, request is now made to the next prior entity in a parallel manner and the algorithm repeats until the size of S_{temp} equals that of primal problem. The merit of GSA cannot be used efficiently in the worst case where the requesting entity set is less satisfied than the responding entity set. In such a situation, the parallel algorithm does not work as expected. UTMOST_SATISFACTION proposes an approach to avoid such scenario and guarantees eradication of worst case for the same problem in future to a maximal extent. This is a variation of MOD_{P-GSA} described above for OPTIMAL_NETWORK_MATCH. The algorithm begins by taking the problem set along with the corresponding solution as input parameter and outputs an optimal matching set □$_i$. For each solution pair in S(A′, B′), we delete each pair and calculate the score of matching formed by the modified preference list. The connection between the pair, whose deletion gives us the minimum score is shut down temporarily which is later assigned the most prior available entity in the preference list. The working of UTMOST_SATISFACTION is explained in detail.

ALGORITHM: UTMOST_SATISFACTION (P, S)

Input: A problem instance P(A, B) and the corresponding solution set, S(A′, B′).

Output: An optimal matching set, η_i.

Precondition: The problem instance, P (A, B) deviates from Satisfiability Rule.

```
Begin
For each pair in solution set, S (A′, B′) do
|    Delete the pair p_i.
|    Set ɱ_i = PARALLEL_GSA (P (A′, B′))
|    Calculate the score, S_m (S_i)
End for
Delete the pair p_i, for which S_m is minimum.
Output the corresponding matching set ɱ_i.
End
```

Lemma 1: *For an instance of 3-sided cyclic network with ties and incomplete lists, PARALLEL_GSA completes with a worst case time complexity of* $|A||B|-2|B|+\log_2|A|$.

Proof: PARALLEL_GSA follows a divide and conquer approach. Lines 2-4 divide the whole problem of size |A| in a binary way and hence will take $\log_2|A|$ number of steps to do so until the P_{size} becomes 1. Lines 5-17 conquer the problem by merging the solutions. Except for the first merging step if we assume that at each merging level a requesting entity falls into the rejected set, R_s, then merging procedure will take |A-2| + (|A-2||B-1|) number of steps, which gives the worst case time complexity PARALLEL_GSA to be $|A||B|-2|B|+\log_2|A|$.

Lemma 2: *For a solution instance which deviates from the Satisfiability Rule, UTMOST_SATISFACTION algorithm is initiated which takes maximum* $|A|*(|A||B|-2|B|+$

$log_2|A|$) number of steps to provide an optimal solution less prone to give a worst case in future.

Proof: OPTIMAL_NETWORK_MATCH (P) sends control to UTMOST_SATISFACTION (P,S) whenever *Satisfiability Rule (Sat (A) > Sat (B))* is deviated. According to GSA, matching set always contains |A| number of matches, where A is the requesting entity set. For loop through line 2-6 runs for each match in S(A′, B′) i.e |A| times. At line 4 a call to PARALLEL_GSA has been made which takes $|A||B|-2|B|+ \log_2|A|$ number of steps according to Lemma 1. Therefore, our algorithm for UTMOST_SATISFACTION (P, S) takes $|A|*(|A||B|-2|B|+ \log_2|A|)$ in total to run.

Theorem: *For an instance of 3-sided cyclic network with ties and incomplete lists with equal number of entities in each set U, D and S i.e. |U| = |D| =|S|, OPTIMAL_NETWORK_MATCH (P) completes and takes at most $o(n^3)$ time.*

Proof: OPTIMAL_NETWORK_MATCH (P) runs in sets of If-Else. For every instance where Satisfiability Rule is broken (line 2, 7 and 12), ULTIMATE_SATISFACTION is called which takes $o(n^3)$ time (from lemma 1), else (line 4, 9 and 14) PARALLEL_GSA is called taking $o(n^2)$ time (from lemma 2). Therefore, the total time complexity of the algorithm is given by $o(n^3)$ for equal number of user, sensor and server entities.

Next chapter describes the experimental set up for finding out the results to our proposed approach.

6. EXPERIMENTAL SETUP

We consider the scenarios of a three-sided network under the restricted model, i.e., the preference lists of users are derived from a master list on data sources and all users are indifferent to the servers. The capacity of servers and all the preference lists, including the ranks and ties, are generated randomly. Normally, the number of servers is fewer than the number of users and it is the capacity of servers that is critical in real applications. We set 2-3 system models while the capacity of each server is varied. A requirement is imposed for the systems must have iperf software installed which is a tool to measure network performance. Iperf was orginally developed by NLANR/DAST as a modern alternative for measuring TCP and UDP bandwidth performance. Iperf is a tool to measure maximum TCP bandwidth, allowing the tuning of various parameters and UDP characteristics. Iperf reports bandwidth, delay jitter, datagram loss. For a TCP connection type, it measures bandwidth, reports MSS/MTU size and observed read sizes. It also support for TCP window size via socket buffers. Multi-threaded if pthreads or Win32 threads are available, Client and server can have multiple simultaneous connections which paves way for parallel connections to be accepted. For a UDP connection, client can create UDP streams of specified bandwidth, measure packet loss and delay jitter. It is multicast capable. Here appropriate, options can be specified with K (kilo-) and M (mega-) suffices, so 128K instead of 131072 bytes. It can run for specified time, rather than a set amount of data to transfer picking the best units for the size of data being reported. The server handles multiple connections, rather than quitting after a single test and prints periodic, intermediate bandwidth, jitter, and loss reports at specified intervals. The server can be run as a daemon as well as Windows NT Service. Moreover we can use representative streams to test out how link layer compression affects our achievable bandwidth

7. RESULTS

The time complexity of both the algorithms MOD_{GSA} and $MOD_{P\text{-}GSA}$ is given by $O(n^3)$ and $o(n^3)$ when calculated theoretically. Taking the worst case scenario as input and the number of steps required as the parameter here we have done a theoretical comparative analysis, which has been later verified programmatically through simulations using the python language. The basic classes taken are:

```
class Player():

    """

    Class for the general player in a matching game

    """

    def __init__(self, name, preferences):

        self.free = True

        self.partner = False

        self.preferences = preferences

        self.name = name


class Suitor(Player):

    """

    Class for the suitors (men) in a matching game.

    """

    def propose(self):

        """
```

```
            Method that returns top reviewer's name from list of
            preferences.

        """

        return self.preferences[0]


class Reviewer(Player):

"""

Class for the reviewers (women) in a matching game.

"""

    def accept_proposal(self, suitor):

     """

    Method that returns True or False depending on whether a
    potential suitor is still in the preference list.

     """

     if suitor.name in self.preferences:

        return True

     return False


class Matching_Game():

"""

Class for a matching game.

"""

def __init__(self, suitor_preferences, reviewer_preferences):
```

```
"""
Initialise
"""
    self.suitor_preferences = suitor_preferences
    self.reviewer_preferences = reviewer_preferences
    self.suitors = sorted(suitor_preferences.keys())
    self.reviewers = sorted(reviewer_preferences.keys())


 def solve(self):
 """
Apply the Extended Gale Shapley Algorithm as described in Irving 1994
 """
self.stable_matching=Gale_Shapley_algorithm(self.suitors,
self.suitor_preferences,self.reviewers,self.reviewer_preference)
def invert_solve(self):
 """
 Apply solving algorithm but swap suitors and reviewers
 """
self.inverted_stable_matching =
Gale_Shapley_algorithm(self.reviewers,
self.reviewer_preferences, self.suitors,
self.suitor_preferences)
```

The worst case scenario is tested for occurrence by calculation the scores for both sides of the matching set. If the worst case occurs to happen the algorithm is run to eradicate it. Deducing the performance enhancement for $MOD_{P\text{-}GSA}$ in comparison to MOD_{GSA} and we conclude that as the value of *n* increases the performance enhancement metric goes on giving better results as shown in table 9.

TABLE 9: COMPARATIVE ANALYSIS

NUMBER OF PEOPLE IN EACH SET (N)	NUMBER OF STEPS			PERFORMANCE ENHANCEMENT IN $MOD_{P\text{-}GSA}$ W.R.T. MOD_{GSA}
	GSA	**MOD_{GSA}**	**$MOD_{P\text{-}GSA}$**	
3	9	12	6	LOW
4	16	36	20	LOW
5	25	80	55	LOW
6	36	150	108	INTERMEDIATE
7	49	252	189	INTERMEDIATE
8	64	392	304	INTERMEDIAE
9	81	576	468	INTERMEDIATE
10	100	810	670	INTERMEDIATE
11	121	1100	924	HIGH
12	144	1452	1236	HIGH
13	169	1872	1612	HIGH
14	196	2366	2058	HIGH
15	225	2940	2184	HIGH
16	256	3600	3184	HIGH

Representing the data from table 9 in a graphical form we obtain fig. 30 showing the comparison of performance for each algorithm. Here also we can see that the difference between the peak points for MOD_{GSA} and $MOD_{P\text{-}GSA}$ keeps on increasing as the value of n increases.

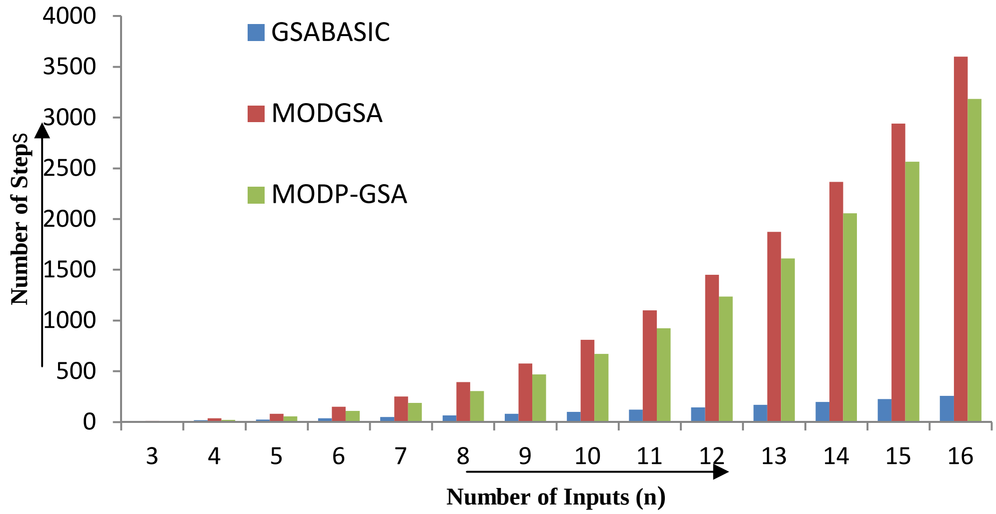


Figure 30: Comparison graph considering 'No. of Steps' as a parameter

We know that the number of steps required for an algorithm to run is directly proportional the time it will take to run on any machine. When we run the algorithm for various problem instances we obtained the graph given in fig 31. The graph shows the variation in time complexities for MOD_{GSA} and $MOD_{P\text{-}GSA}$ and the edge $MOD_{P\text{-}GSA}$ obtains over MOD_{GSA} for larger values of n.

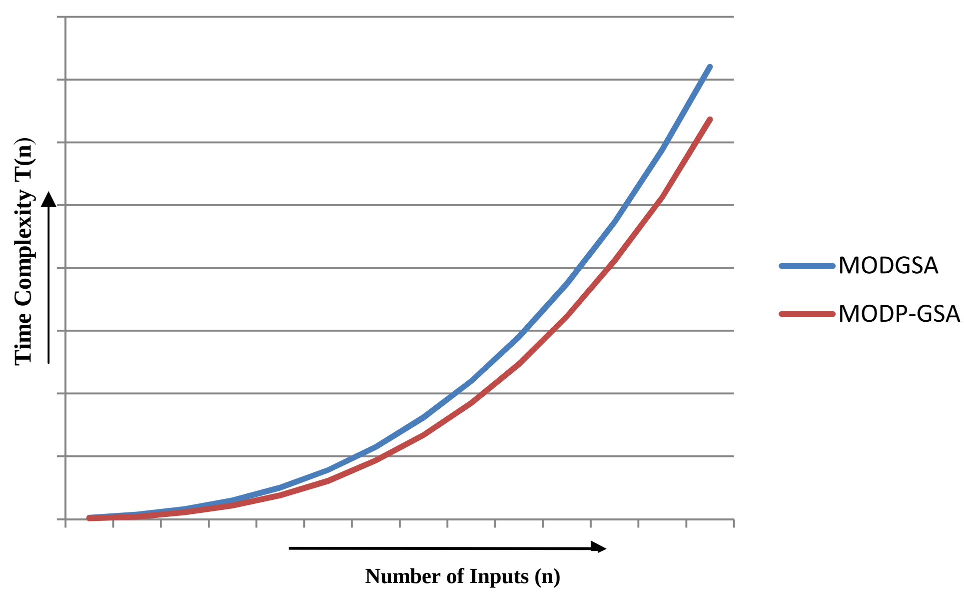


Figure 31: Comparison of Time Complexities of MOD_{GSA} AND $MOD_{P\text{-}GSA}$

As a summary our first part of our work explores the worst case scenario of Gale Shapely Algorithm (GSA) and improves it by deleting one pair to achieve greater happiness. For this I have proposed an algorithm, MOD_{GSA}, based on GSA_{BASIC} which takes $O(n^3)$ time. We have taken a number of steps to run the algorithm as our parameter and represented our results both in tabular and graphical form. However, on comparing we deduced the result that $MOD_{P\text{-}GSA}$ gives better performance than MOD_{GSA} for higher values of n and hence achieving greater stability.

Extending our algorithms for networks, we have proposed OPTIMAL_NETWORK_MATCH we have set up a network in our university with 4 users, 4 sensors and 3 servers, which has been later done for two other models. We explore the case of three-sided matching where the users, servers and data sources show an interesting cyclic preference. The details are described as follows:

1. Users only have preference on the data sources based on their requirements. The preference list of u_i is defined as a subset of data sources that are prioritized according to service quality they provide, e.g., the preference list $P(u_i)$ contains all sensors within the requested area, ranking according to their accuracy or distance to the target location. In $P(u_i)$, data sources with the same priority form a tie in the list. Normally, users do not care about from which server the requested data is received. Any servers can forward data to users, as long as the corresponding pairs are acceptable.
2. Servers only have preference on users when receiving service requests. The preference list of s_j is $P(s_j)$ $\{u_i|u_j \in U, bij >= \varphi\}$. The order can be identified through the cost of each user, e.g., the requested stream rate. A server normally will forward data for any data sources (these data sources are determined by the users that the server decides to serve), as long as the corresponding pair is acceptable. In $P(s_j)$, users with the same priority form a tie in the list.
3. Data sources usually receive service requests of users directly from servers. They only need to choose which server the data should be delivered to. So, data sources only have preference on servers. The preference list of d_k is $P(d_k)$ $\{s_j|s_j \in S, b_{jk} >= \varphi\}$.

The order can be determined by the connection quality, e.g., link delays, loss ratio. In $P(d_k)$, servers with the same priority form a tie in the list. Thus, the preference relationship among data sources, servers and users are cyclic. The objective of the system is to find an optimal matching on the three types of agents, in which users choose data sources, data sources choose servers and servers choose users, satisfying their preference. The measured prioritizing parameters are given in table 10(a) and 10(b).

TABLE 10: PRIORITIZING PARAMETERS FOR STABLE MATCHING

USER	SENSOR	STRENGTH	$DEMAND_{BIT\_RATE}$ (KBPS)	$DATA_{SIZE}$ (BYTES)
u1	d1	4	96	63,914,238
u1	d2	4	96	63,914,238
u2	d1	4	500	7,193,808
u2	d2	5	500	7,193,808
u3	d4	3	800	49,624,521
u4	d1	3	300	258,099,247
u4	d3	4	300	258,099,247

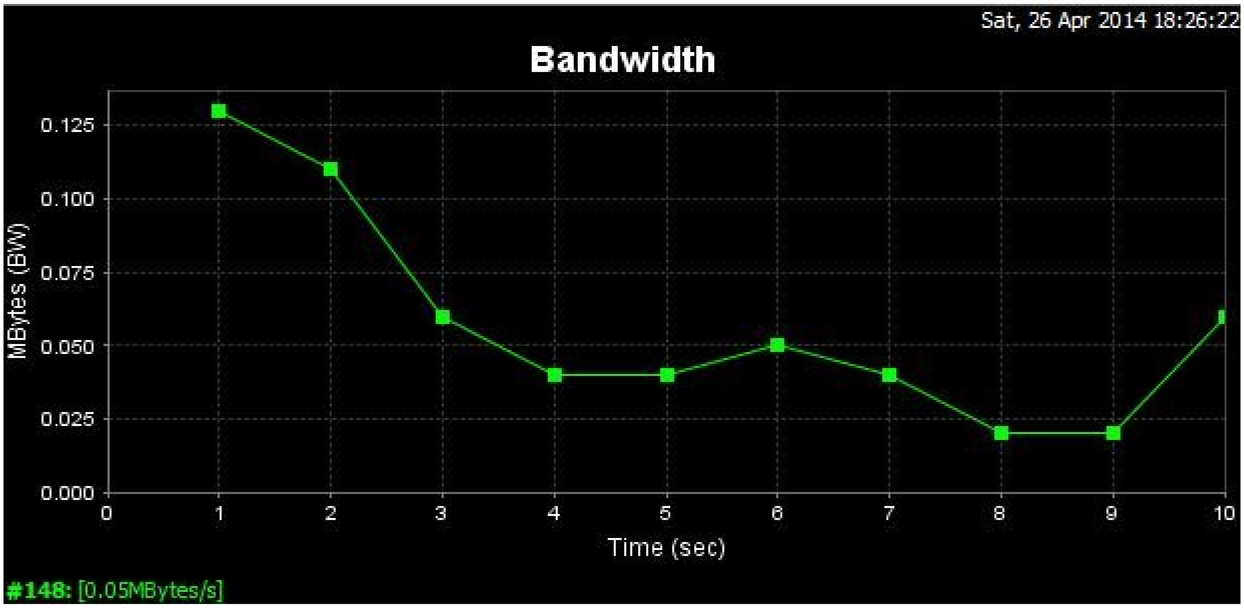
(a)

SENSOR	SERVER	CAPACITY	LINK_DIST (MTRS)	BANDWIDTH (MBPS)	PD (MS)
d1	s1	1	60	100	20
d1	s3	1	150	95	20
d1	s2	2	200	72	20
d2	s2	2	75	78	25
d2	s1	1	80	56	25
d3	s3	1	98	100	17
d3	s2	2	100	100	17
d3	s1	1	200	95	17
d4	s3	1	50	75	15
d4	s2	2	75	56	15

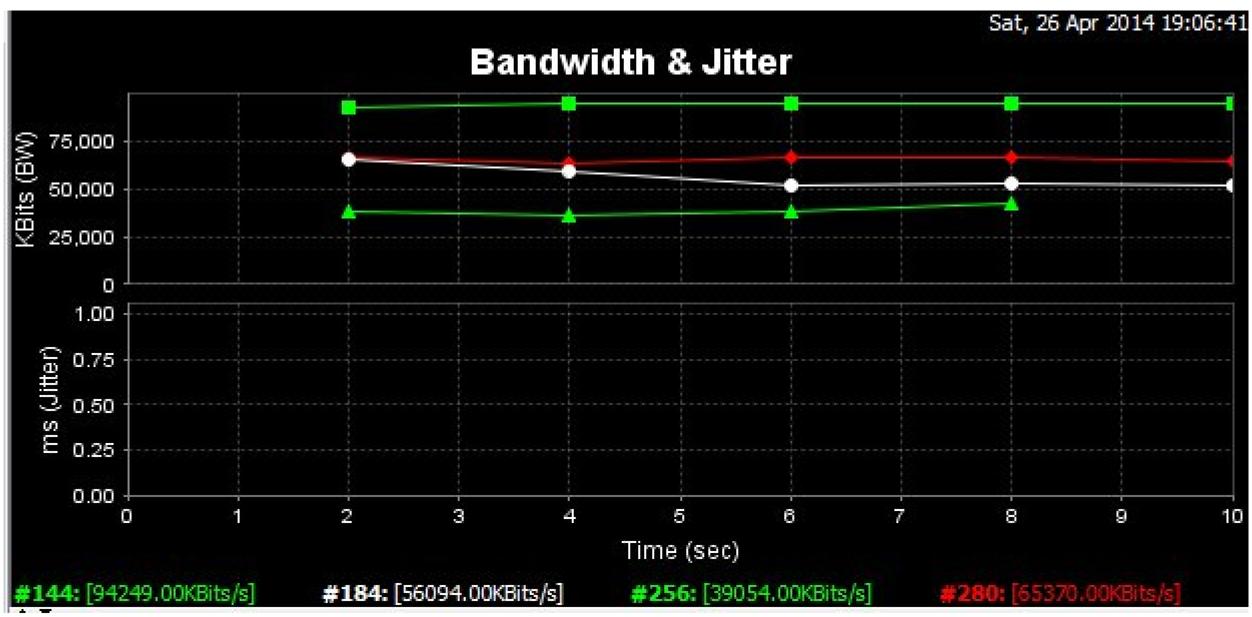
(b)

Table 10 (a) consists of the information users (u1, u2, u3, u4) need to provide as input. Strength is represented in numeric form (out of 5) as the number of vertical lines seen for each user-sensor pair. Based upon this criteria alone users preference list for sensors is formed. $Demand_{Bit\text{-}rate}$ is the required bitrate that server must provide to transfer file of size $Data_{size}$. Any server failing to pass this criterion stands ineligible to provide service to the user through the specific sensor. Table 10 (b) lists the parameters between the available sensor-server pairs maintained by the servers. Each server is limited for the number of users it can provide service to, based upon the configuration on the server. Here, we have the 's2' which can provide service to 2 users, rest of the two servers have capacity 1 each. '*Link_dist*' is the distance between any sensor-server pair measured in unit of meters. Bandwidth is the rate at which data can be sent from the server to the sensor which varies for each sensor-server pair. Propagation delay (P_d) is specific to each sensor and is measured in milliseconds. Now considering these parameters as input by the server we calculate the propagation time from Eq. (1) using Eq. (2), Eq(3) and Eq(4). P_t is the prioritizing criterion for generating the preference list for sensors.

To generate the servers table with users in its preference list, the prioritizing criterion is the requested stream rate of the users. As server always tries to provide services to greater number of users, the lower the $Demand_{Bitrate}$ in the users-sensors table in table 11 (a), more preferred the user is. Another restrictive factor in determining the preference list in servers table is the available bandwidth between the users and servers. To determine this we are using the *iperf* software. Output window of this software for the user (u1) and the server (s1) is shown in Fig 32 (a) and (b) respectively.



(a)



(b)

Fig 32. Output from iperf software for an user and server determining the available bandwidth and jitter between user-server pair

With the prioritizing parameters we get the resultant users, sensors and servers table as shown in fig 33, with the red cells marked as the matching partner for each table.

U_ID	PREF_1	PREF_2	PREF_3
u1	d1	d2	
u2	d2	d1	d4
u3	d4	d3	
u4	d3	d1	

D_ID	PREF_1	PREF_2	PREF_3
d1	s1	s3	s2
d2	s2	s1	
d3	s3	s2	s1
d4	s3	s2	

S_ID	PREF_1	PREF_2	PREF_3
d1	s1	s3	s2
d2	s2	s1	
d3	s3	s2	s1
d4	s3	s2	

Fig 33. Users (U), Sensors (D) and Server (S) Table

Simulation has been done in PHP & MySQL for the algorithm OPTIMAL_NETWORK_MATCH. With the given input, we find out the possible matches by using the query:

```
$query1 =

”CREATE TABLE matches1 AS SELECT

usr1.u_user AS A, ssr1.d_sensor AS B, svr1.s_server AS C,
usr2.u_user AS D FROM users AS usr1 INNER JOIN sensors AS ssr1 ON
(    usr1.u_sensor1 = ssr1.d_sensor OR

     usr1.u_sensor2 = ssr1.d_sensor OR

     usr1.u_sensor3 = ssr1.d_sensor OR

     usr1.u_sensor4 = ssr1.d_sensor

)

INNER JOIN servers AS svr1 ON

(     ssr1.d_server1 = svr1.s_server OR

      ssr1.d_server2 = svr1.s_server OR

      ssr1.d_server3 = svr1.s_server

)
```

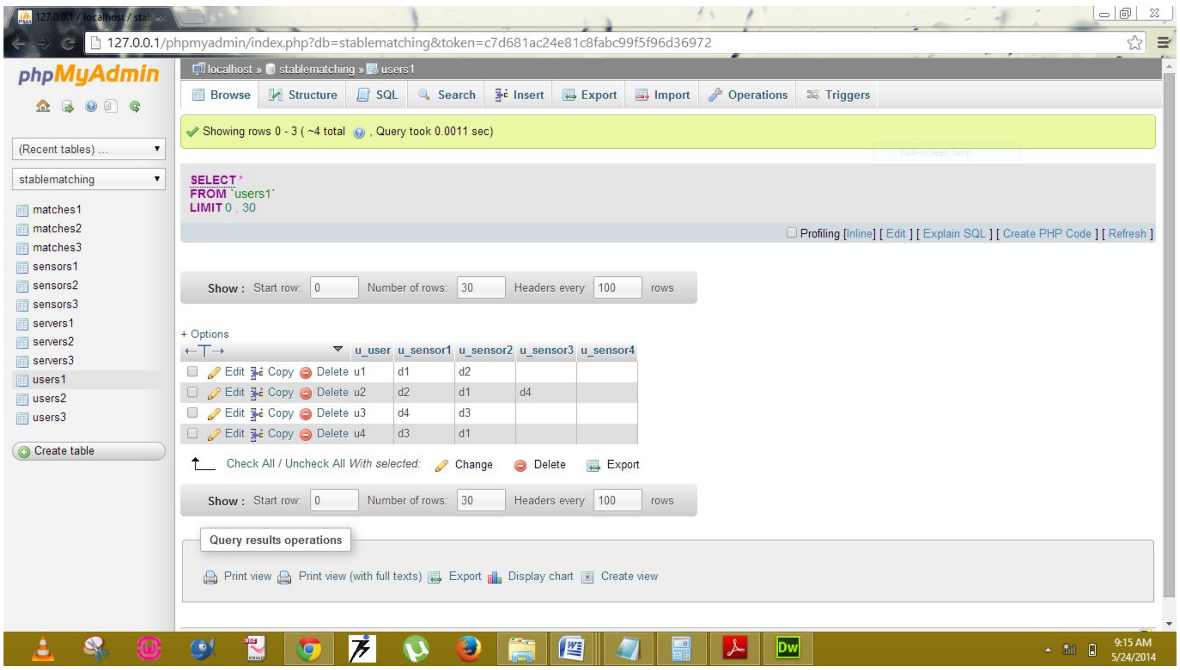
```
INNER JOIN users AS usr2 ON
(     svr1.s_user1 = usr2.u_user OR
      svr1.s_user2 = usr2.u_user OR
      svr1.s_user3 = usr2.u_user OR
      svr1.s_user4 = usr2.u_user
)
WHERE usr1.u_user = usr2.u_user ORDER BY A, B, C;”
```

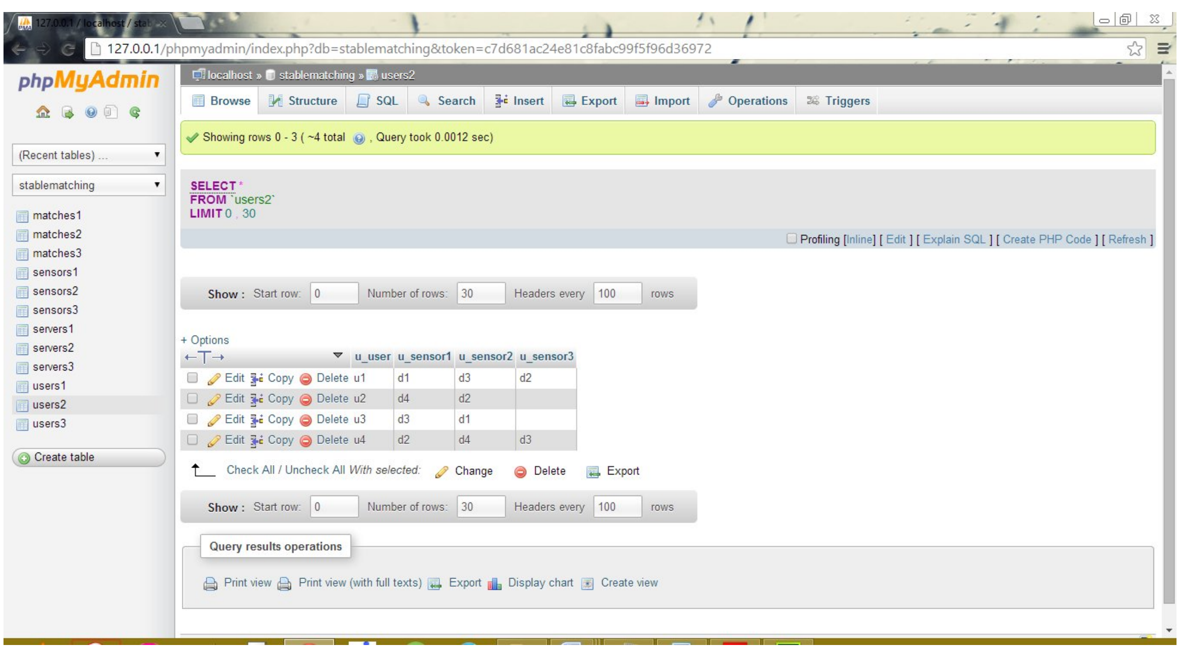
TABLE 11: POSSIBLE MATCHES TABLE

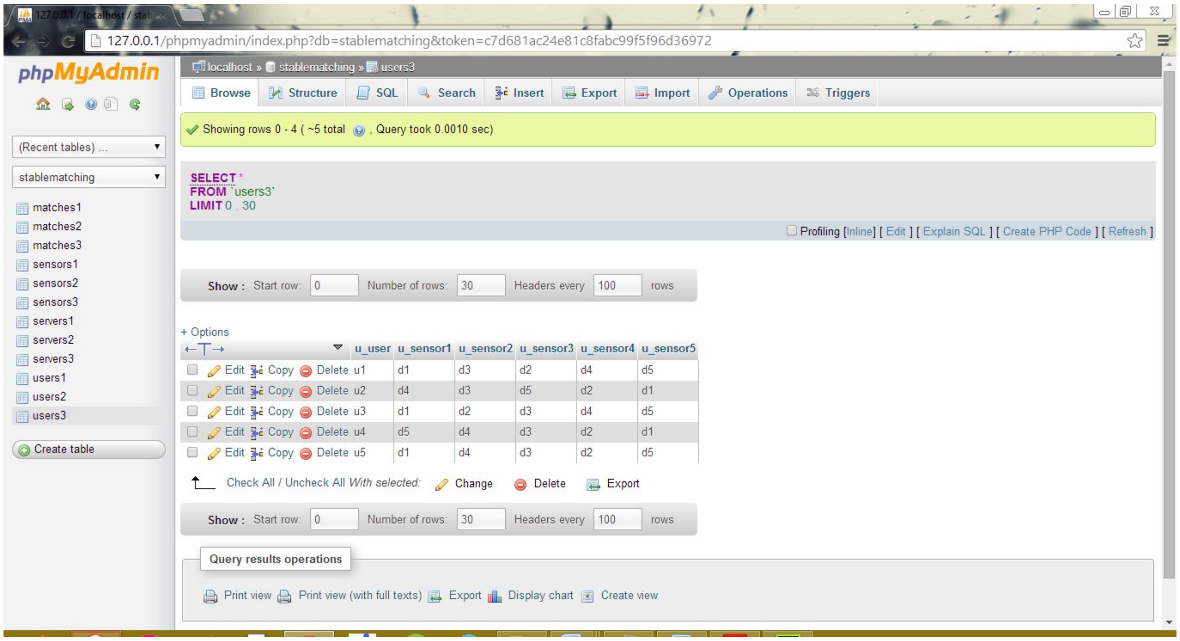
U_ID	D_ID	S_ID	U_ID	HAPPINESS
u_1	d_1	s_1	u_1	3
u_1	d_2	s_1	u_1	5
u_2	d_1	s_1	u_2	5
u_2	d_1	s_2	u_2	7
u_2	d_2	s_1	u_2	5
u_2	d_2	s_2	u_2	4
u_2	d_4	s_2	u_2	7
u_3	d_3	s_1	u_3	8
u_3	d_3	s_2	u_3	5
u_3	d_3	s_3	u_3	5
u_3	d_4	s_2	u_3	4
u_3	d_4	s_3	u_3	4
u_4	d_1	s_1	u_4	7
u_4	d_1	s_2	u_4	8
u_4	d_1	s_3	u_4	5
u_4	d_3	s_1	u_4	8
u_4	d_3	s_2	u_4	6
u_4	d_3	s_3	u_4	3

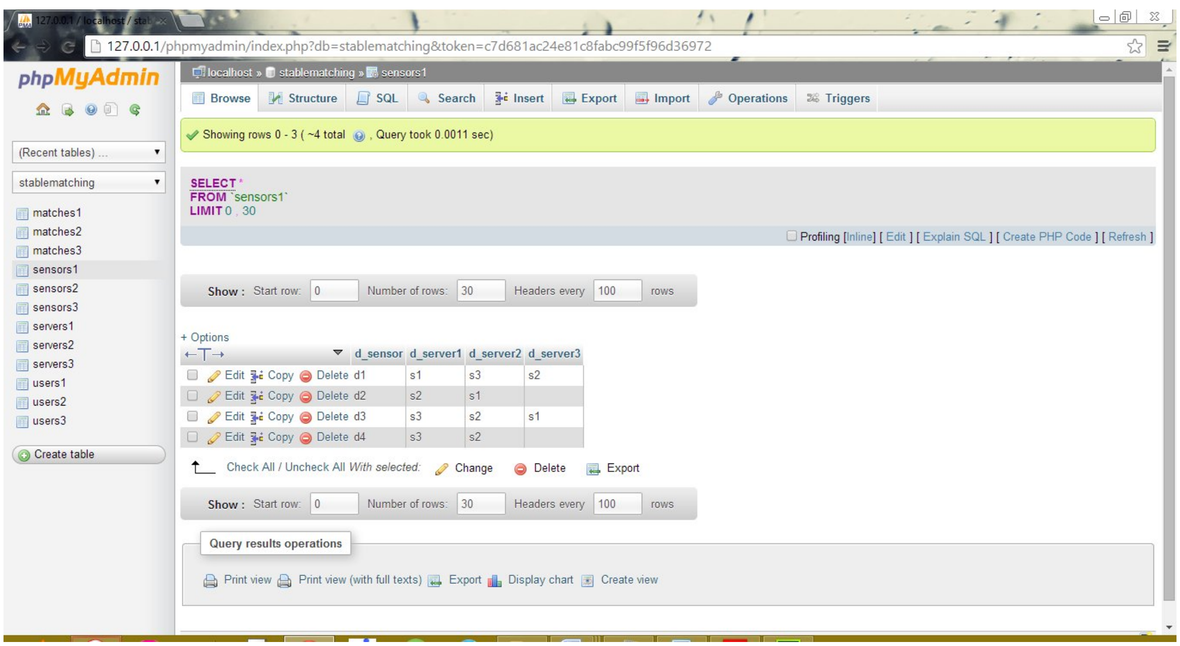
This query results to output the table shown in table 11. From this table, now our job is just to filter it for storing the rows with minimum happiness score for each user. The snapshots of the MySQL tables and the output screen has been shown for 3 problem instances out of which the third instance is a NP Hard problem.

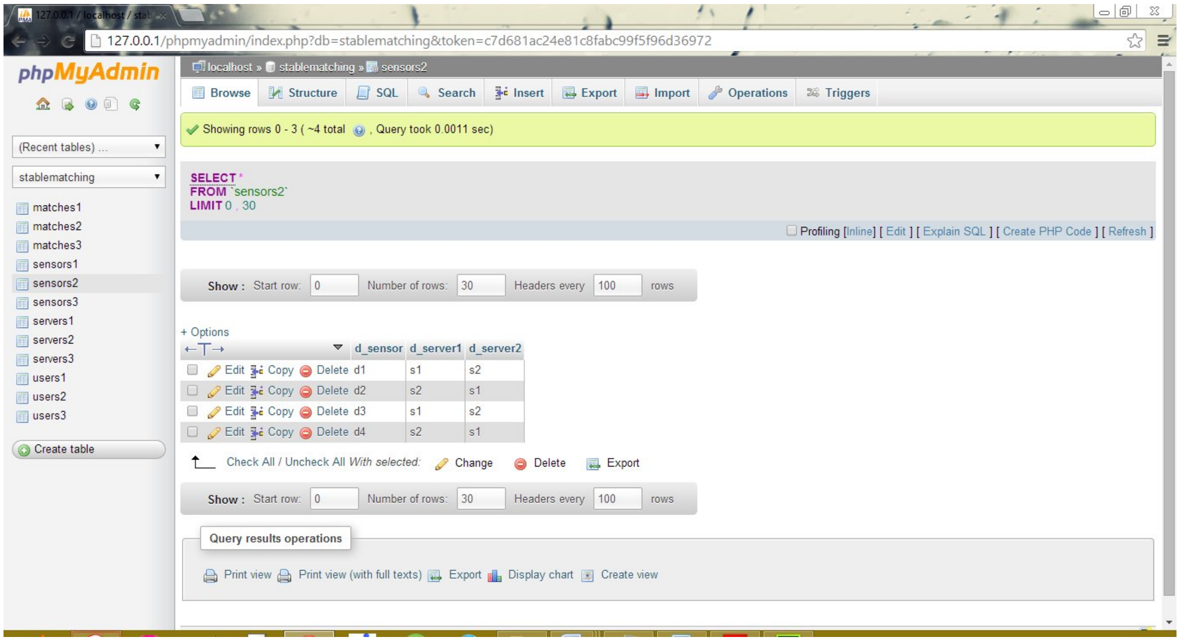
MYSQL SNAPSHOTS

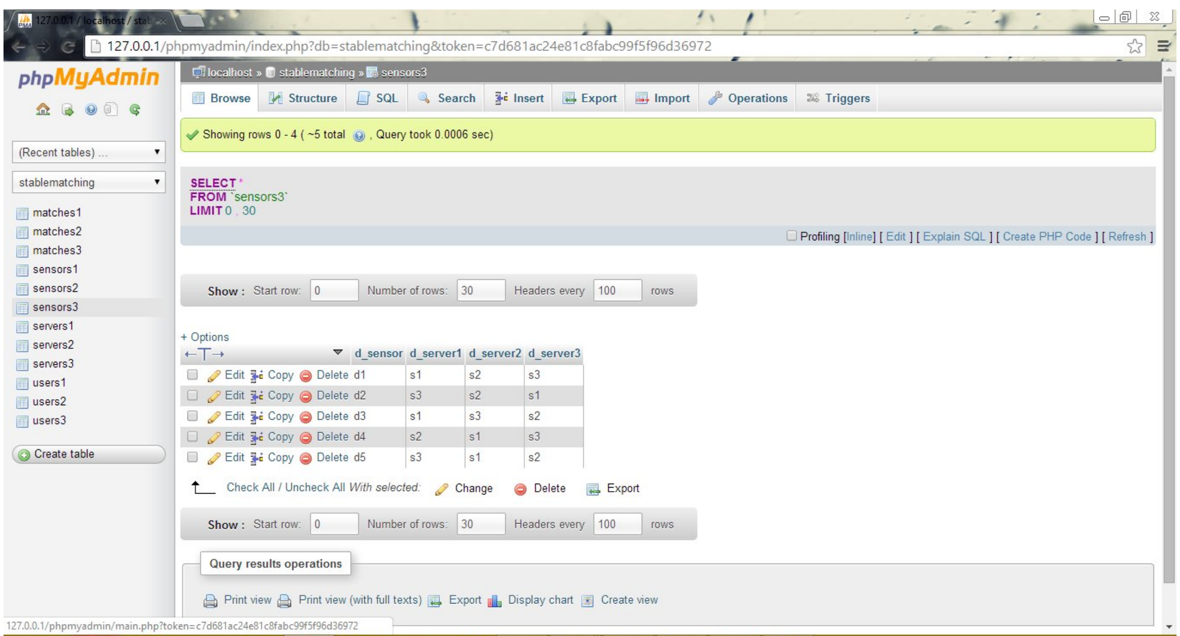


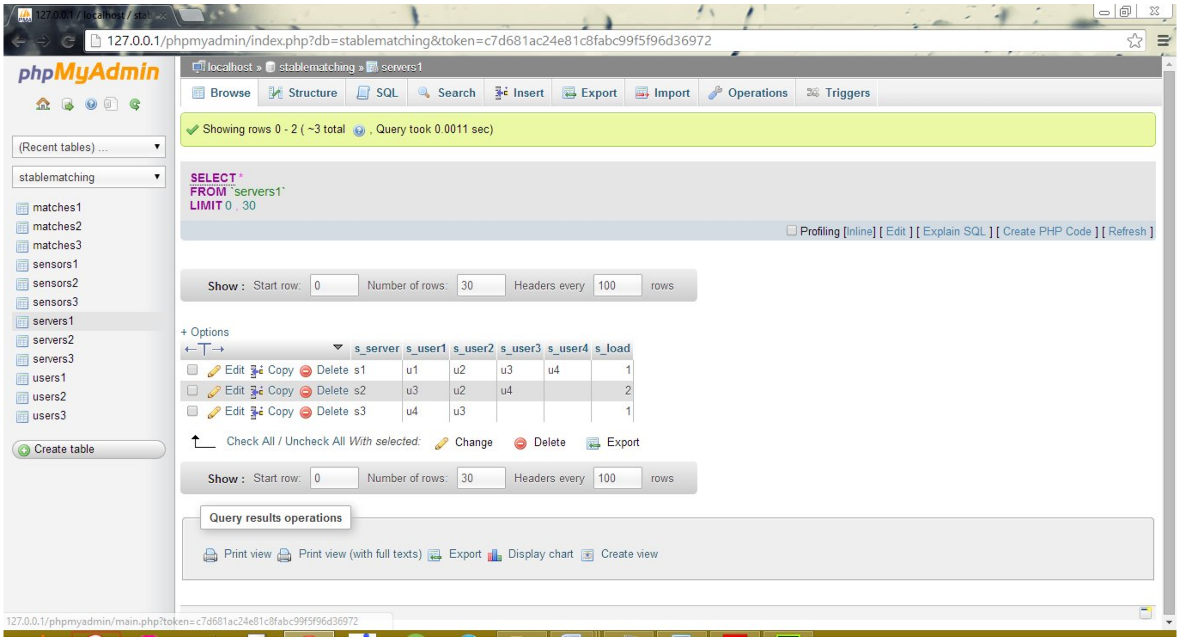


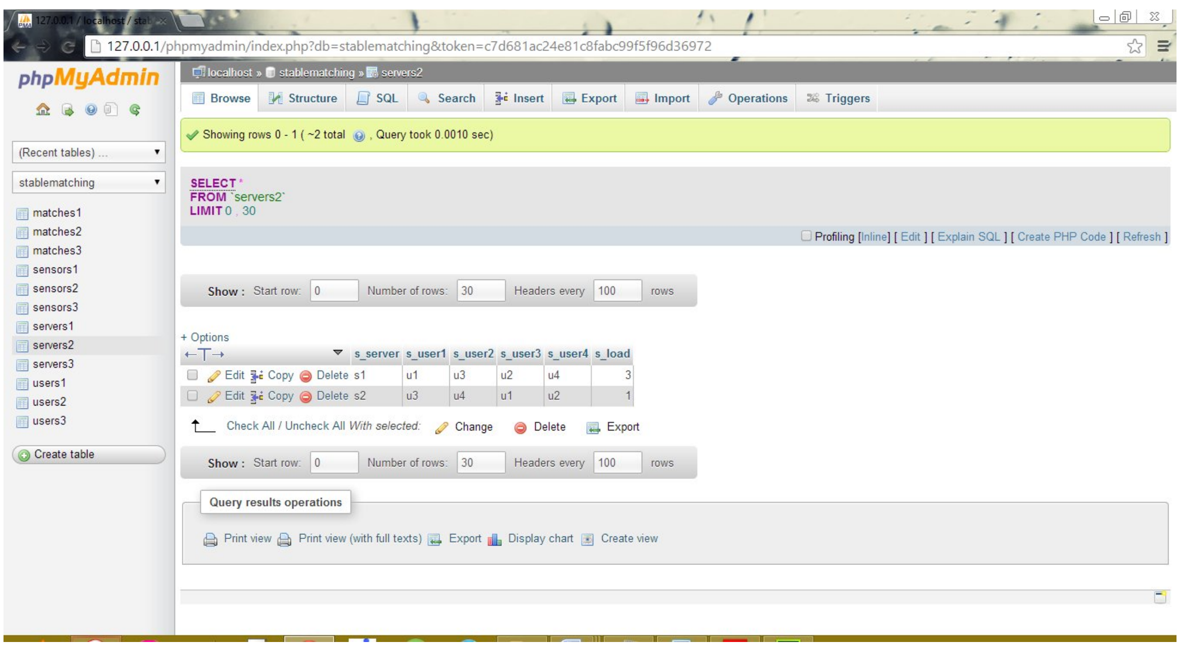

127.0.0.1/phpmyadmin/index.php?db=stablematching&token=c7d681ac24e81c8fabc99f5f96d36972
phpMyAdmin
localhost » stablematching » users3
Browse Structure SQL Search Insert Export Import Operations Triggers
Showing rows 0 - 4 (~5 total , Query took 0.0010 sec)
SELECT *
FROM `users3`
LIMIT 0 , 30
Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]
(Recent tables) ...
stablematching
matches1
matches2
matches3
sensors1
sensors2
sensors3
servers1
servers2
servers3
users1
users2
users3
Create table
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
+ Options
u_user u_sensor1 u_sensor2 u_sensor3 u_sensor4 u_sensor5
Edit Copy Delete u1 d1 d3 d2 d4 d5
Edit Copy Delete u2 d4 d3 d5 d2 d1
Edit Copy Delete u3 d1 d2 d3 d4 d5
Edit Copy Delete u4 d5 d4 d3 d2 d1
Edit Copy Delete u5 d1 d4 d3 d2 d5
Check All / Uncheck All With selected: Change Delete Export
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
Query results operations
Print view Print view (with full texts) Export Display chart Create view

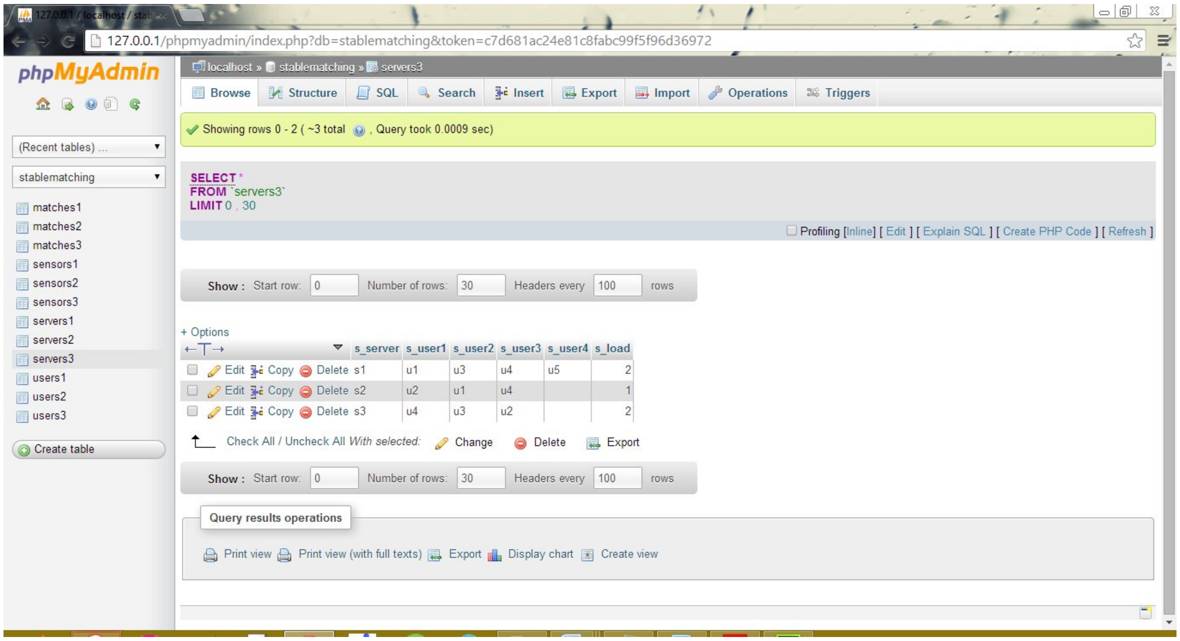

127.0.0.1/phpmyadmin/index.php?db=stablematching&token=c7d681ac24e81c8fabc99f5f96d36972
phpMyAdmin
localhost » stablematching » sensors1
Browse Structure SQL Search Insert Export Import Operations Triggers
Showing rows 0 - 3 (~4 total , Query took 0.0011 sec)
SELECT *
FROM `sensors1`
LIMIT 0 , 30
Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]
(Recent tables) ...
stablematching
matches1
matches2
matches3
sensors1
sensors2
sensors3
servers1
servers2
servers3
users1
users2
users3
Create table
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
+ Options
d_sensor d_server1 d_server2 d_server3
Edit Copy Delete d1 s1 s3 s2
Edit Copy Delete d2 s2 s1
Edit Copy Delete d3 s3 s2 s1
Edit Copy Delete d4 s3 s2
Check All / Uncheck All With selected: Change Delete Export
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
Query results operations
Print view Print view (with full texts) Export Display chart Create view

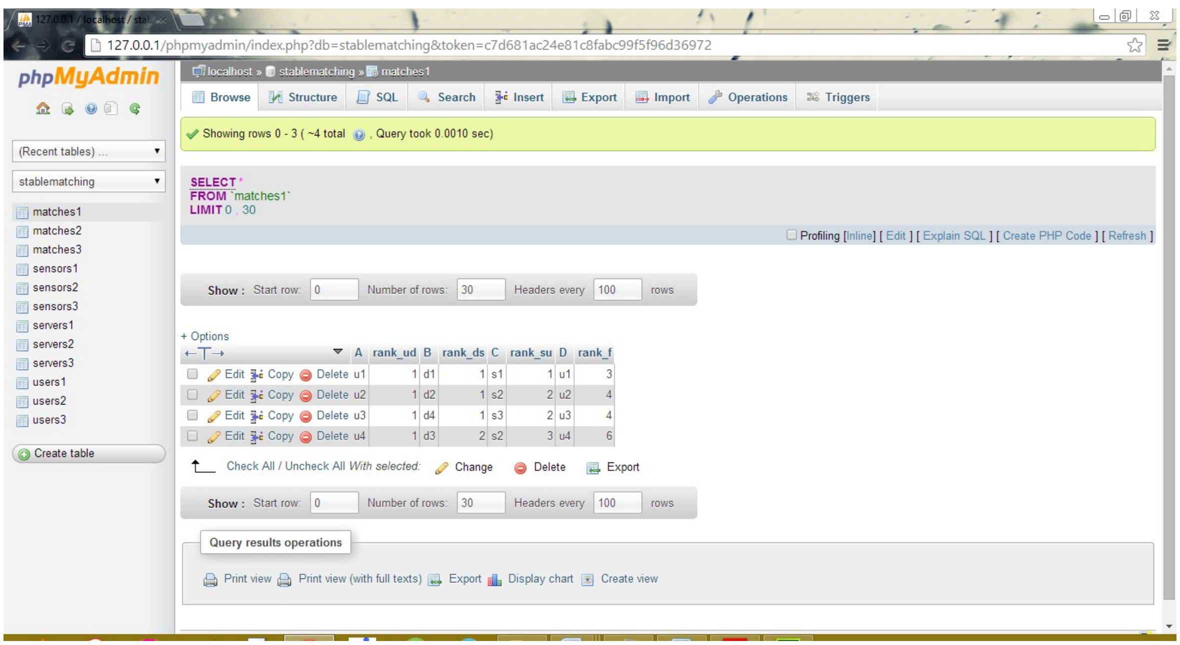

127.0.0.1/phpmyadmin/index.php?db=stablematching&token=c7d681ac24e81c8fabc99f5f96d36972
phpMyAdmin
localhost » stablematching » sensors2
Browse Structure SQL Search Insert Export Import Operations Triggers
Showing rows 0 - 3 (~4 total , Query took 0.0011 sec)
SELECT *
FROM `sensors2`
LIMIT 0 , 30
Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]
(Recent tables) ...
stablematching
matches1
matches2
matches3
sensors1
sensors2
sensors3
servers1
servers2
servers3
users1
users2
users3
Create table
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
+ Options
d_sensor d_server1 d_server2
Edit Copy Delete d1 s1 s2
Edit Copy Delete d2 s2 s1
Edit Copy Delete d3 s1 s2
Edit Copy Delete d4 s2 s1
Check All / Uncheck All With selected: Change Delete Export
Query results operations
Print view Print view (with full texts) Export Display chart Create view

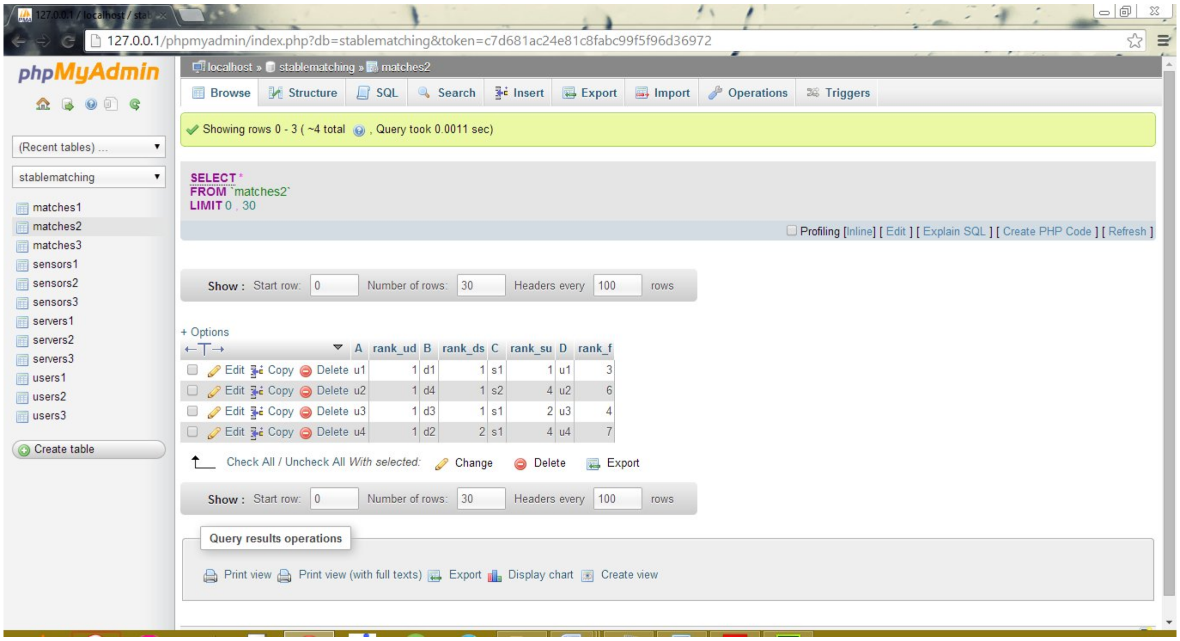

127.0.0.1/phpmyadmin/index.php?db=stablematching&token=c7d681ac24e81c8fabc99f5f96d36972
phpMyAdmin
localhost » stablematching » sensors3
Browse Structure SQL Search Insert Export Import Operations Triggers
Showing rows 0 - 4 (~5 total , Query took 0.0006 sec)
SELECT *
FROM `sensors3`
LIMIT 0 , 30
Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]
(Recent tables) ...
stablematching
matches1
matches2
matches3
sensors1
sensors2
sensors3
servers1
servers2
servers3
users1
users2
users3
Create table
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
+ Options
d_sensor d_server1 d_server2 d_server3
Edit Copy Delete d1 s1 s2 s3
Edit Copy Delete d2 s3 s2 s1
Edit Copy Delete d3 s1 s3 s2
Edit Copy Delete d4 s2 s1 s3
Edit Copy Delete d5 s3 s1 s2
Check All / Uncheck All With selected: Change Delete Export
Query results operations
Print view Print view (with full texts) Export Display chart Create view
127.0.0.1/phpmyadmin/main.php?token=c7d681ac24e81c8fabc99f5f96d36972

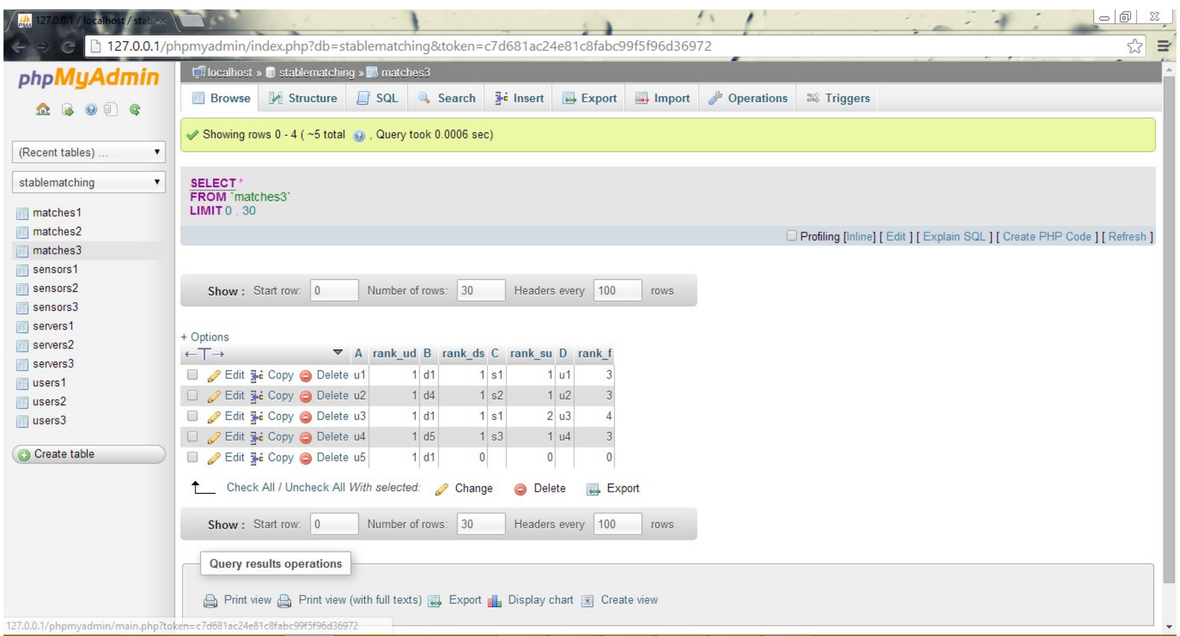

127.0.0.1/phpmyadmin/index.php?db=stablematching&token=c7d681ac24e81c8fabc99f5f96d36972
phpMyAdmin
localhost » stablematching » servers1
Browse Structure SQL Search Insert Export Import Operations Triggers
Showing rows 0 - 2 (~3 total , Query took 0.0011 sec)
SELECT *
FROM `servers1`
LIMIT 0 , 30
Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]
(Recent tables) ...
stablematching
matches1
matches2
matches3
sensors1
sensors2
sensors3
servers1
servers2
servers3
users1
users2
users3
Create table
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
+ Options
s_server s_user1 s_user2 s_user3 s_user4 s_load
Edit Copy Delete s1 u1 u2 u3 u4 1
Edit Copy Delete s2 u3 u2 u4 2
Edit Copy Delete s3 u4 u3 1
Check All / Uncheck All With selected: Change Delete Export
Query results operations
Print view Print view (with full texts) Export Display chart Create view


127.0.0.1/phpmyadmin/index.php?db=stablematching&token=c7d681ac24e81c8fabc99f5f96d36972
phpMyAdmin
localhost » stablematching » servers2
Browse Structure SQL Search Insert Export Import Operations Triggers
Showing rows 0 - 1 (~2 total , Query took 0.0010 sec)
SELECT *
FROM `servers2`
LIMIT 0 , 30
Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]
(Recent tables) ...
stablematching
matches1
matches2
matches3
sensors1
sensors2
sensors3
servers1
servers2
servers3
users1
users2
users3
Create table
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
+ Options
s_server s_user1 s_user2 s_user3 s_user4 s_load
Edit Copy Delete s1 u1 u3 u2 u4 3
Edit Copy Delete s2 u3 u4 u1 u2 1
Check All / Uncheck All With selected: Change Delete Export
Query results operations
Print view Print view (with full texts) Export Display chart Create view

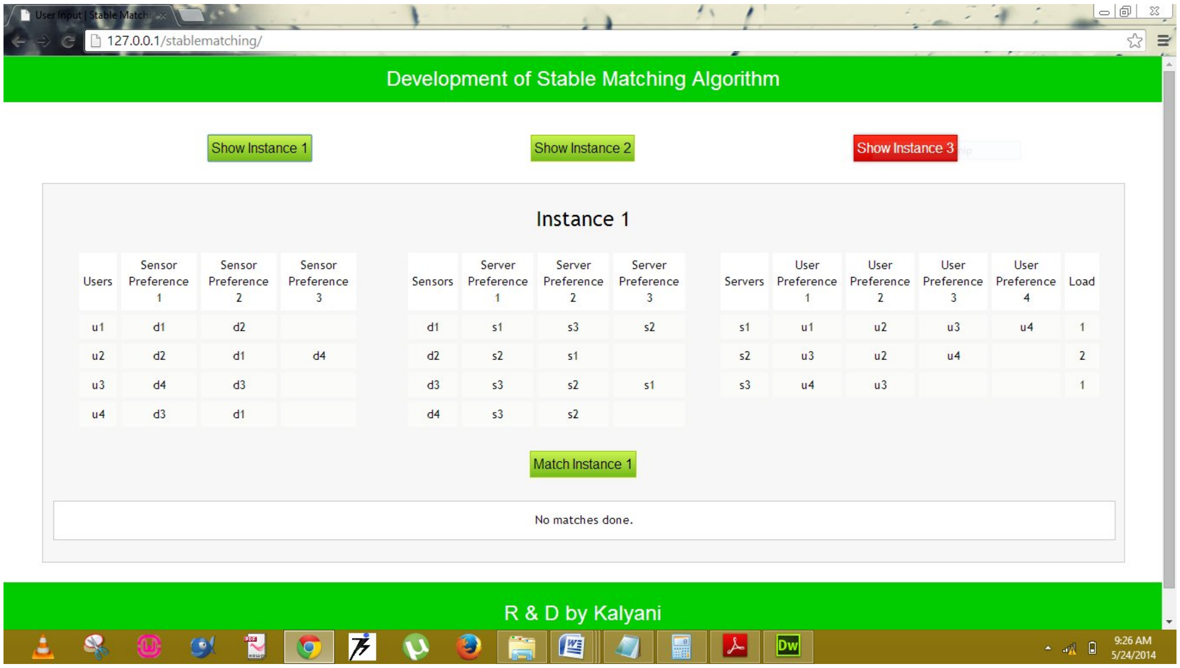

127.0.0.1/phpmyadmin/index.php?db=stablematching&token=c7d681ac24e81c8fabc99f5f96d36972
phpMyAdmin
localhost » stablematching » servers3
Browse Structure SQL Search Insert Export Import Operations Triggers
Showing rows 0 - 2 (~3 total , Query took 0.0009 sec)
SELECT *
FROM `servers3`
LIMIT 0 , 30
Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]
(Recent tables) ...
stablematching
matches1
matches2
matches3
sensors1
sensors2
sensors3
servers1
servers2
servers3
users1
users2
users3
Create table
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
+ Options
s_server s_user1 s_user2 s_user3 s_user4 s_load
Edit Copy Delete s1 u1 u3 u4 u5 2
Edit Copy Delete s2 u2 u1 u4 1
Edit Copy Delete s3 u4 u3 u2 2
Check All / Uncheck All With selected: Change Delete Export
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
Query results operations
Print view Print view (with full texts) Export Display chart Create view

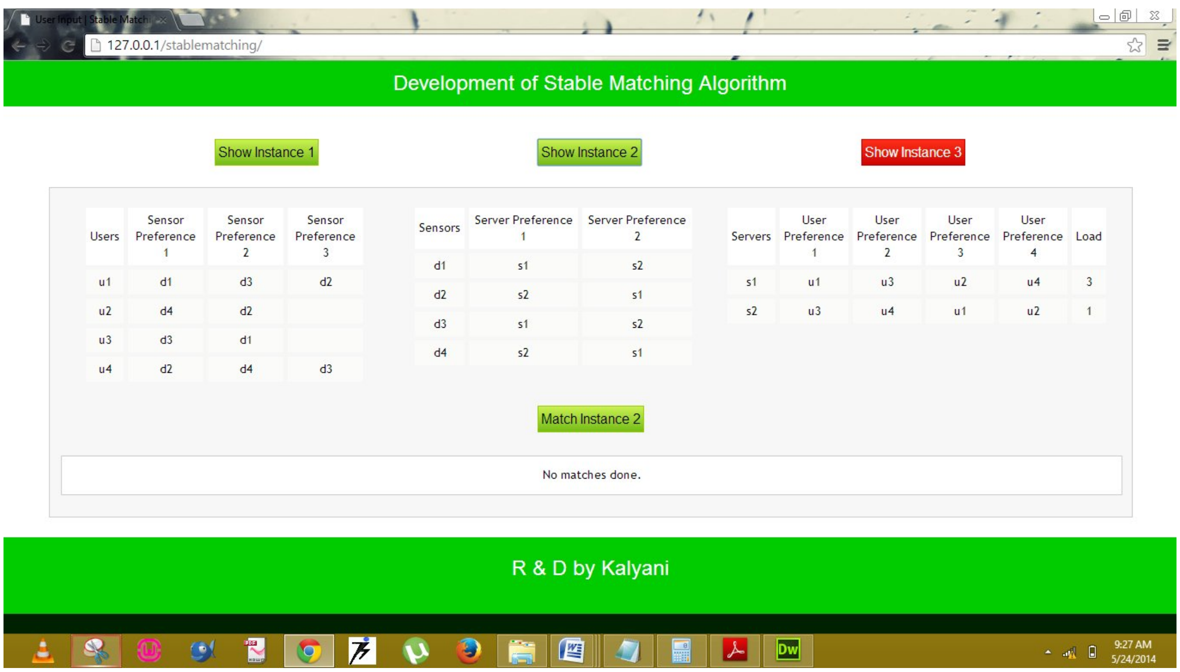

127.0.0.1/phpmyadmin/index.php?db=stablematching&token=c7d681ac24e81c8fabc99f5f96d36972
phpMyAdmin
localhost » stablematching » matches1
Browse Structure SQL Search Insert Export Import Operations Triggers
Showing rows 0 - 3 (~4 total , Query took 0.0010 sec)
SELECT *
FROM `matches1`
LIMIT 0 , 30
Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]
(Recent tables) ...
stablematching
matches1
matches2
matches3
sensors1
sensors2
sensors3
servers1
servers2
servers3
users1
users2
users3
Create table
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
+ Options
A rank_ud B rank_ds C rank_su D rank_f
Edit Copy Delete u1 1 d1 1 s1 1 u1 3
Edit Copy Delete u2 1 d2 1 s2 2 u2 4
Edit Copy Delete u3 1 d4 1 s3 2 u3 4
Edit Copy Delete u4 1 d3 2 s2 3 u4 6
Check All / Uncheck All With selected: Change Delete Export
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
Query results operations
Print view Print view (with full texts) Export Display chart Create view

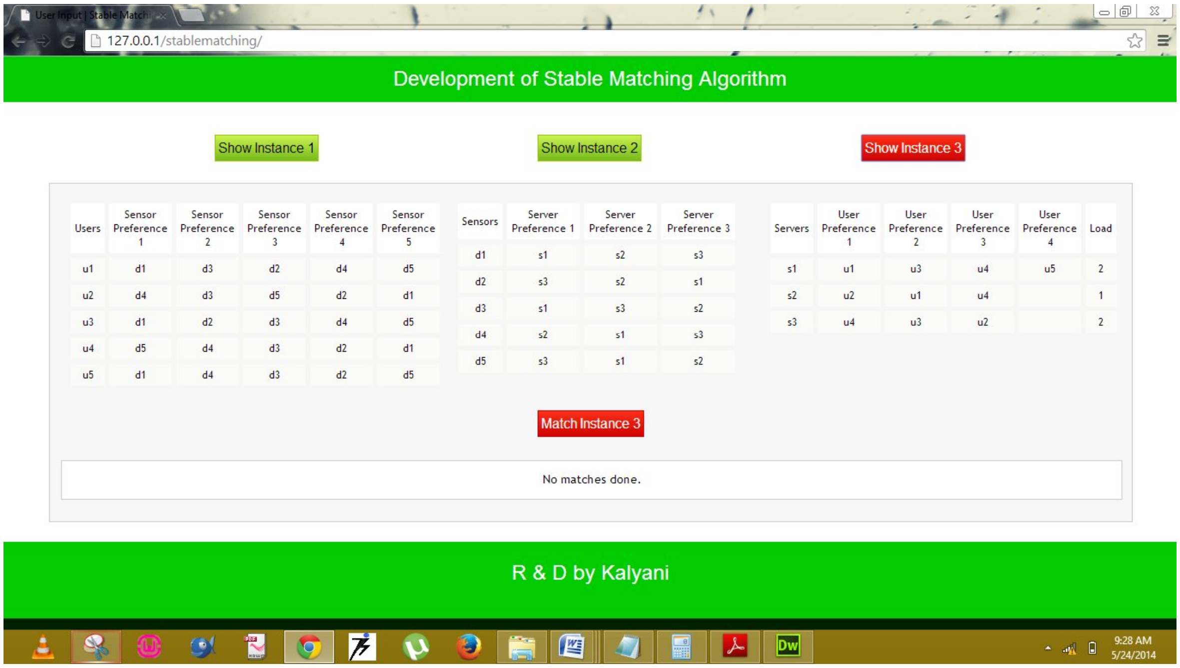

127.0.0.1/phpmyadmin/index.php?db=stablematching&token=c7d681ac24e81c8fabc99f5f96d36972
phpMyAdmin
(Recent tables) ...
stablematching
matches1
matches2
matches3
sensors1
sensors2
sensors3
servers1
servers2
servers3
users1
users2
users3
Create table
localhost » stablematching » matches2
Browse Structure SQL Search Insert Export Import Operations Triggers
Showing rows 0 - 3 (~4 total , Query took 0.0011 sec)
SELECT *
FROM `matches2`
LIMIT 0 , 30
Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
+ Options
A rank_ud B rank_ds C rank_su D rank_f
Edit Copy Delete u1 1 d1 1 s1 1 u1 3
Edit Copy Delete u2 1 d4 1 s2 4 u2 6
Edit Copy Delete u3 1 d3 1 s1 2 u3 4
Edit Copy Delete u4 1 d2 2 s1 4 u4 7
Check All / Uncheck All With selected: Change Delete Export
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
Query results operations
Print view Print view (with full texts) Export Display chart Create view


127.0.0.1/phpmyadmin/index.php?db=stablematching&token=c7d681ac24e81c8fabc99f5f96d36972
phpMyAdmin
(Recent tables) ...
stablematching
matches1
matches2
matches3
sensors1
sensors2
sensors3
servers1
servers2
servers3
users1
users2
users3
Create table
localhost » stablematching » matches3
Browse Structure SQL Search Insert Export Import Operations Triggers
Showing rows 0 - 4 (~5 total , Query took 0.0006 sec)
SELECT *
FROM `matches3`
LIMIT 0 , 30
Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
+ Options
A rank_ud B rank_ds C rank_su D rank_f
Edit Copy Delete u1 1 d1 1 s1 1 u1 3
Edit Copy Delete u2 1 d4 1 s2 1 u2 3
Edit Copy Delete u3 1 d1 1 s1 2 u3 4
Edit Copy Delete u4 1 d5 1 s3 1 u4 3
Edit Copy Delete u5 1 d1 0 0 0
Check All / Uncheck All With selected: Change Delete Export
Show : Start row: 0 Number of rows: 30 Headers every 100 rows
Query results operations
Print view Print view (with full texts) Export Display chart Create view

OUTPUT SNAPSHOTS



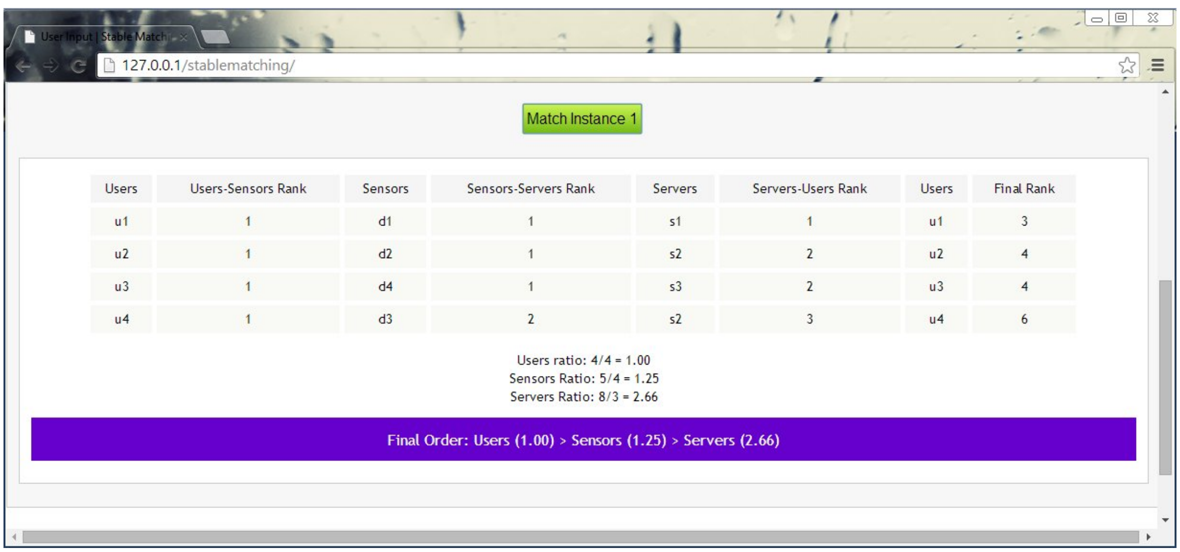




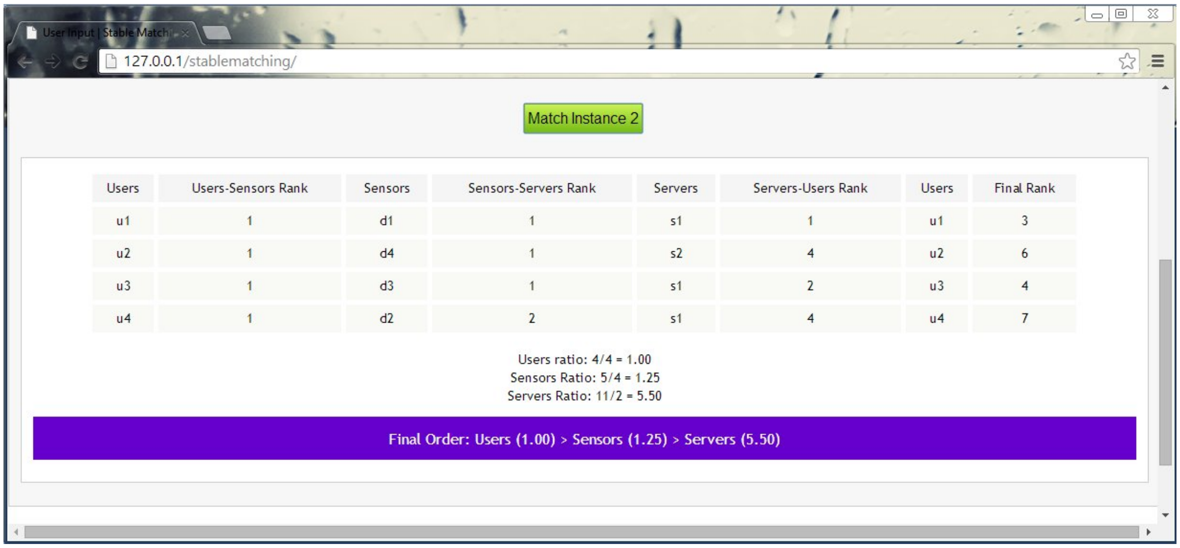
Users	Sensor Preference 1	Sensor Preference 2	Sensor Preference 3	Sensor Preference 4	Sensor Preference 5
u1	d1	d3	d2	d4	d5
u2	d4	d3	d5	d2	d1
u3	d1	d2	d3	d4	d5
u4	d5	d4	d3	d2	d1
u5	d1	d4	d3	d2	d5

Sensors	Server Preference 1	Server Preference 2	Server Preference 3
d1	s1	s2	s3
d2	s3	s2	s1
d3	s1	s3	s2
d4	s2	s1	s3
d5	s3	s1	s2

Servers	User Preference 1	User Preference 2	User Preference 3	User Preference 4	Load
s1	u1	u3	u4	u5	2
s2	u2	u1	u4		1
s3	u4	u3	u2		2



Users	Users-Sensors Rank	Sensors	Sensors-Servers Rank	Servers	Servers-Users Rank	Users	Final Rank
u1	1	d1	1	s1	1	u1	3
u2	1	d2	1	s2	2	u2	4
u3	1	d4	1	s3	2	u3	4
u4	1	d3	2	s2	3	u4	6



User Input | Stable Match

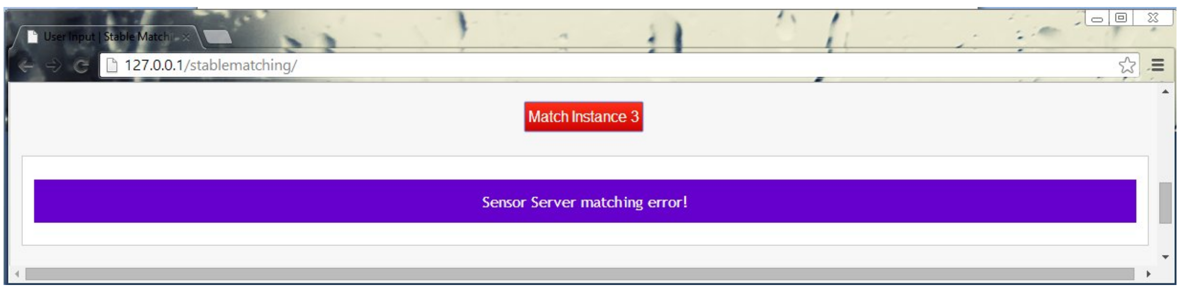
127.0.0.1/stablematching/

Match Instance 2

Users	Users-Sensors Rank	Sensors	Sensors-Servers Rank	Servers	Servers-Users Rank	Users	Final Rank
u1	1	d1	1	s1	1	u1	3
u2	1	d4	1	s2	4	u2	6
u3	1	d3	1	s1	2	u3	4
u4	1	d2	2	s1	4	u4	7

Users ratio: 4/4 = 1.00
Sensors Ratio: 5/4 = 1.25
Servers Ratio: 11/2 = 5.50

Final Order: Users (1.00) > Sensors (1.25) > Servers (5.50)



User Input | Stable Match

127.0.0.1/stablematching/

Match Instance 3

Sensor Server matching error!

8. CONCLUSION

Our work has explored the worst case scenario of Gale Shapely Algorithm (GSA) and improves it by deleting one pair to achieve greater happiness. For this we have proposed an algorithm, MOD_{GSA}, based on GSA_{BASIC} which takes $O(n^3)$ time. Further, we have been trying to decrease this time complexity, by following the parallel GSA ($GSA_{PARALLEL}$), denoted by $MOD_{P\text{-}GSA}$. We have taken a number of steps to run the algorithm as our parameter and represented our results both in tabular and graphical form. However, on comparing we deduced the result that $MOD_{P\text{-}GSA}$ gives better performance than MOD_{GSA} for higher values of n and hence achieving greater stability.

This report also proposed an algorithm to optimally make a stable match between a group of users, sensors and severs with cyclic preferences using the concept of parallel stable matching with incomplete lists and ties. Algorithmic details along with the lemmas prove the completeness of the algorithm with a time complexity of $o(n^3)$. Experimental results show the dynamic nature of generating the preference list for each entity type and application of algorithm to generate the output.

Future scope of this application includes extension of OPTIMAL_NETWORK_MATCH to distributed environment where a single sensor can handle multiple requests. Failure of the server providing services leads to starvation situation for the paired user. This could have been avoided by mirroring the servers or initiating communication between the servers, leading to a possible application in the field of shared data access with multiple sensors requests.

REFERENCES

[1] D. Gale, L.S. Shapley 1962 College admissions and the stability of marriage, The American Mathematical Monthly 69, pp. 9–15.

[2] Takao Inoshita, Robert W. Irvin, Kazuo Iwama, Shuichi Miyazaki and Takash Nagase 2013 Improving Man optimality Stable Matching by Minimum Change of Preference Lists, Algorithms, 6, 371-382; doi:10.3390/a6020371

[4] An extension of stable matching, G Demange, D Gale, M Sotomayor - Discrete Applied Mathematics, (1987) – Elsevier

[5] Huang, C.-C. Cheating by Men in Gale Shapley Stable Matching Algorithm. In the proceeding of ESA 2006, Zurich, Switzerland, 11-13 September (2006); pp. 418-431

[6] S.S. Tseng and R.C.T. Lee, A Parallel Algorithm to solve the Stable Marriage Problem, BIT 24:308, 316 (1984)

[7] Michael J. Quinn, A note on Two parallel Algorithms to solve the stable marriage problem, BIT 25:473, (1985)

[8] Jesper Larsen, A Parallel Approach to Stable Marriage Problem, (1997)

[9] David J. Abraham1 and Telikepalli Kavitha, Dynamic Matching Markets and Voting Paths. 10th Scandinavian Workshop on Algorithm Theory, Riga, Latvia, July 6-8, 2006. Proceedings. pp 65-76. DOI 10.1007/11785293_9.

[10] Ismel Brito and Pedro Meseguer, Distributed Stable Marriage Problem, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006. Proceedings, pp DOI: 675-679 10.1007/11889205_49

[11] National Resident Matching Program, http://www.nrmp.org/

[12] Scottish PRHO Allocations, http://www.dcs.gla.ac.uk/˜rwi/SPA.html

[13] Japan Residency Matching Program, http://www.jrmp.jp/

[14] T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch, "Strongly stable matchings in time O(nm) and extension to the H/R problem," Proc. STACS 2004, pp. 222–233, 2004.

[15] K. Iwama, S. Miyazaki, N. Yamauchi, "A (2 − c√1N)- approximation algorithm for the stable marriage problem," Proc. ISAAC 2005, LNCS 3827, pp. 902-914.

[16] K. Iwama, S. Miyazaki, N. Yamauchi. “A 1.875- approximation algorithm for the stable marriage problem”. In the Proceeding of SODA 2007. pp. 288–297.

[17] Huang, “Cheating by Men in Gale Shapley Stable Matching Algorithm”. In the proceeding of ESA 2006, Zurich, Switzerland, 11-13 September (2006); pp. 418-431

[18] R. W. Irving, “An efficient algorithm for the “stable roommates” problem,” J. Algorithms, Vol. 6, No. 4, pp. 577–595, 1985

[19] Eytan Ronn, “Np-complete stable matching problems”, Journal of Algorithms **11** (1990), no. 2, 285-304.

[20] R. W. Irving and D. F. Manlove, “The stable roommates problem with ties,” J. Algorithms, Vol. 43, No. 1, pp. 85–105, 2002.

[21] Tam´as Fleinera, Robert W. Irvingb, David F. Manloveb “Efficient algorithms for generalized Stable Marriage and Roommates problems” Theoretical Computer Science 381 (2007) 162–176

[22] S. Scott, “A study of stable marriage problems with ties,” Ph.D. Thesis, University of Glasgow, 2005.

[23] Ronn E., NP-complete stable matching problems. Journal of Algorithms (1990), 11:285–304.

[24] Roth E., and Sotomayor M.A.O., “Two-sided matching: A study in game-theoretic modeling and analysis” (1992),, Cambridge Univ Press,I SBN 0521437881.

[25] Peter Biro and Sofa Kiselgof. College admissions with stable score- limits. Central European Journal of Operations Research (2013). pp. 1- 15. DOI: 10.1007/s10100-013-0320-9

[26] A. Romero-Medina. Implementation of stable solutions in a restricted matching market. Review of Economic Design, 3(2):137-147, 1998.

[27] M. Balinski and T. Sonmez. A tale of two mechanisms: Student placement. Journal of Economic Theory, 84(1):73-94, 1999.

[28] Rapaport FT: The case for a living emotionally related international kidney donor exchange registry. Transplant Proc 18: 5–9, 1986

[29] Park, K. Lee, J. H., Huh, K. H., Kim,Y. S., "Exchange Living Donor Kidney Transplantation: Diminution of Donor Organ Shortage" (2004), Transplantation Proceedings, 36, 2949–2951

[30] M. Utku Unver. Dynamic Kidney Exchange. Review of Economic Studies (2010) 77, 372–414. doi: 10.1111/j.1467-937X.2009.00575.x

[31] Alvin E. Roth, The evolution of the labor market for medical interns and residents: A case study in game theory, Journal of Political Economy 92 (1984), no. 6, 991.

[32] Tayfun Sonmez. Housing Markets & Top Trading Cycles. 16th Jerusalem Summer School in Economic Theory "Matching, Auctions, and Market Design"

[33] P´eter Bir´o and Eric McDermid. Three-sided stable matchings with cyclic preferences. Algorithmica, this issue. DOI:10.1007/s00453-009-9315-2, 2009.

[34] Knuth, D.E.: Mariages Stables. Les Presses de L'Université de Montréal, Montréal (1976)

[35] Alkan, A.: Non-existence of stable threesome matchings. Math. Soc. Sci. 16, 207–209 (1988)

[36] Ng, C., Hirschberg, D.S.: Three-dimensional stable matching problems. SIAM J. Discrete Math. 4(2), 245–252 (1991)

[37] Subramanian, A.: A new approach to stable matching problems. SIAM J. Comput. 23(4), 671–700 (1994)

[38] Endre Boros, Vladimir Gurvich, Steven Jaslar, and Daniel Krasner. Stable matchings in three-sided systems with cyclic preferences. Discrete Mathematics, 289(1-3):1–10, 2004.

[39] Eriksson, K., Sjöstrand, J., Strimling, P.: Three-dimensional stable matching with cyclic preferences. Math. Soc. Sci. 52(1), 77–87 (2006)

[40] Huang, C.-C.: Two's company, three's a crowd: stable family and threesome roommate problems. In the proceeding of algorithm ESA lecture notes in computational science., vol. 4698, pp. 558–569. Springer, Berlin (2007)

[41] Péter Biró, Eric McDermid. Three-Sided Stable Matchings with Cyclic Preferenes. In the journal of algorithmica, 2010. Volume 58, Issue 1, pp 5-18 . DOI: 10.1007/s00453-009-9315-2

[42] A. Subramanian, The computational complexity of the circuit value and network stability problems, Ph.D. Thesis, Department of Computer Science, Stanford University, 1990.

[43] A. Subramanian, A new approach to stable matching problems, SIAM Journal of Computing 23 (4) (1994) 671–700.

[44] A. Subramanian, Nitin, On a performance of multistage interconnection network, in: Proceedings of the IEEE ADCOM, December 15–18, 2004, pp. 73–79.

[45] Nitin, Ashok Subramanian, Efficient algorithms and methods to solve dynamic MINs stability problem using stable matching with complete ties. Journal of Discrete Algorithms 6 (2008) 353–380. doi:10.1016/j.jda.2008.01.001

[46] Prabhakar B, and McKeown N, "On the speedup required for combined input and output queued switching,"Computer Systems Lab, Stanford Univ., Stanford, CA, Tech. Rep. CSL-TR-97-738. 1999

[47] Robert W. Irving. An efficient algorithm for the stable roommates problem. J. Algorithms, 6:577–595, 1985.

[48] Dan Gusfield and Robert W. Irving. The Stable Marriage Problem: Structure and Algorithms. The MITPress, Cambridge, Massachusetts, 1989

[49] Boris Pittel and Robert W. Irving. An upper bound for the solvability of a random stable roommates instance. Random Struct. Algorithms, 5(3):465–487, 1994.

[50] P.Erdos¨ and A. Ren´ yi. On the evolution of randomgraphs. Magyar Tud.Akad. Mat. Kutato´ Int. Kozl ¨ , 5:17–61, 1960.

[51] Stephan Mertens, Random Stable Matchings, Journal of Statistics Mechanism. (2005) pp 1- 11

[52] David J. Abraham and Telikepalli Kavitha, Dynamic Matching Markets and Voting Paths? In the proceeding of 10th Scandinavian Workshop on Algorithm Theory, Riga, Latvia, July 6-8, 2006. pp 65-76. DOI: 10.1007/11785293_9

[53] Hong Xu, Baochun Li, "Anchor: A Versatile and Efficient Framework for Resource Management in the Cloud", IEEE transaction on parallel and distributed systems, Vol 24: 6, June 2013. Doi: 10.1109/TPDS.2012.308

[54] Siavash Bayat, Raymond H. Y. Louie, Zhu Han, Yonghui Li, and Branka Vucetic, "Multiple Operator and Multiple Femtocell Networks: Distributed Stable Matching". Proc. Wireless Networks Symposium on ICC (IEEE 2012), pp 5140 - 5145

[55] A. Frank, "Augmenting graphs to meet edge-connectivity requirements", Proc. 31st Annual Symposium on Foundations of Computer Science, St. Louis, Oct. 22-24, 1990.

PUBLICATIONS

1. Ekta Gupta, Kalyani and Nitin. Article: Preserving the Basic Property of Stable Matching by Deleting a Pair. IJCA Proceedings on International Conference on Distributed Computing and Internet Technology ICDCIT-2014:14-18. Published by Foundation of Computer Science, New York, USA
2. Kalyani and Nitin. Optimal Network Match: An Application of User-Oriented Stable Matching to 3-Sided Cyclic Networks. Second International Conference on Emerging Research in Computing, Information, Communication and Application ERCICA-2014 $_{\text{[Elsevier India]}}$ (Accepted)