A FRAMEWORK FOR SOLVING HARD VARIANTS OF STABLE MATCHING WITHIN A LIMITED TIME

Tarmo Veskioja
Tallinn Technical University, Department of Informatics
Raja 15, Tallinn, Estonia, 12618

Leo Võhandu
Tallinn Technical University, Department of Informatics
Raja 15, Tallinn, Estonia, 12618

ABSTRACT

In the original stable marriage problem all the participants have to rank all members of the opposite party. Two variations for this problem allow for incomplete preference lists and ties in preferences. Most of the real-world matching problems allow for both types of relaxations. Finding a maximum cardinality solution for the stable matching problem with both ties and incomplete lists (SMTI) is NP-Complete and even the approximation is APX-hard. Finding an egalitarian solution for SMTI is also NP-Complete and APX-hard. If members from one side are allowed to form couples and submit combined preferences, then the set of stable matchings may be empty and determining if a market has any stable matchings is NP-Complete. Real-world applications of centralized matching need to provide a solution within a limited time. We propose a matching framework that always gives a solution. It is based on a genetic algorithm that uses intermediate or approximate solutions from other matching algorithms. We also give a broader definition for the core of marriage game, when the set of stable matchings is empty and it is necessary to use majority voting between matchings in a tournament. We use monotone systems based approach for tournament selection.

KEYWORDS

Stable matching, genetic algorithm, monotone systems, core, majority assignment.

1. INTRODUCTION

Stable marriage problem has attracted a considerable amount of interest after the problem was first formulated by Gale and Shapley (Gale et al., 1962). Many centralized two-sided markets can be described as variants of stable marriage problem. An instance of the original stable marriage problem (SM) consists of N men and N women, with each person having a preference list that totally orders all members of the opposite sex. A man and a woman form a blocking pair in a matching if both prefer each other to their current partners. A matching is stable if it contains no blocking pair. In every instance of SM there is at least one stable matching (Gale et al., 1962) that can be found in $O(N^2)$ time.

One variant of SM allows for both incomplete preferences and ties in the preferences (called SMTI). Finding maximum cardinality matching in SMTI is NP-Complete (Iwama et al., 1999) and even the approximation is APX-hard (Halldórsson et al., 2002). In a Hospital-Residents market (HR), also called one-to-many market, one hospital can provide several positions to the residents. An even more general variant of SM allows for participants of both sides to have a quota (many-to-many market). Each participant can be in a number of pairs, which is limited by his/her quota. This variant is called the stable b-matching problem.

Besides different variants of stable marriage, there is also the question of finding a set of matching criteria that produces fair matchings both at individual level and at the level of parties. Since the "market game" is played on at least two levels - the level of assignments and the level of market rules - it is important to find a stable solution (of rules and matchings) on all levels. Several other matching criteria have been proposed besides different levels of stability and sum of preferences (egalitarian matching).

Matching is a majority assignment (best-voted matching) if there is no other matching that is preferred by a majority of participants to the original matching. Gärdenfors (Gärdenfors, 1975) observed that, when preferences are strict, the set of majority assignments comprises the set of stable matchings, thus showing that the notion of majority assignment is a relaxation of stability (Klijn et al., 2003). Weakly stable matching is a matching, possibly having a blocking pair undermining the stability of a matching, but this blocking pair is not credible in the sense that one of the partners may find a more attractive partner with whom he forms another blocking pair for the original matching (Klijn et al., 2003). In other words, Klijn and Masso define an individually rational matching to be weakly stable if every blocking pair is - in the sense discussed above - not credible. Clearly, weak stability is also a relaxation of stability.

Many markets also require taking into account some additional constraints - for example in HR a pair of residents may have formed a couple and prefer to find a placement at the same hospital, or at least work in the same city (Roth et al., 1990). In this case, the couple submits rank ordered preferences over acceptable pairs of hospitals. After acceptable pairs of hospitals the couple can give rank ordered preferences over single pairs of hospital – couple member, where one of the members of the couple is left without a pair. In this article these mentioned constraints will be called couple constraints.

With most of the real-world matching applications being NP-Complete and APX-hard, a natural question arises how to provide a matching within a limited time. We propose a matching framework that always gives a matching, although it is obviously not always an optimal matching. The framework is based on a genetic algorithm that uses intermediate or approximate matchings from other matching algorithms. The best matchings from each population are chosen by the tournament ranking. We also propose a tournament algorithm based on monotone systems and a value function for it. The proposed algorithm minimizes the number of transitivity faults in the tournament ranking on all tournament tables up to size 5x5. We show how the proposed tournament method works on a model with couple constraints, where voting results between matchings are not transitive. We also define a more relaxed core definition for the marriage models where the set of stable matchings may be empty.

2. MATCHING FRAMEWORK

The proposed framework is based on a genetic algorithm. A genetic algorithm is very easy to implement. It consists of the following basic steps - proper coding of the problem instance, generating initial population of solutions (matchings), using a fitness function to evaluate each solution, finding the best solutions and letting them live, creating offsprings by using crossovers, initiating random mutations in the (genetic) code of the solution.

Besides being able to search the solution space by itself, the genetic algorithm can also accept solutions from other algorithms that solve the same or a similar problem. Not many of matching algorithms can do that. Many approximation and heuristic matching algorithms are candidates to supply the genetic algorithm with good approximate solutions. For example, almost any matching market can be reduced to SM, thus enabling to use the Gale-Shapley algorithm to find one-side optimal solutions to the reduced problem. These solutions are not guaranteed to be optimal to the original problem, but these are good enough for the framework that uses a genetic algorithm. By using solutions from different matching algorithms, that have different complexity phase transitions, a genetic algorithm can minimize the cost of complexity phase transitions. This framework can also make use of parallel computation.

The coding of a stable matching solution should be as simple as possible - a list of men (or a list of women) and their partners. The pairs are ordered by the original order of men in the input data, but the sequential number of men are not shown. If a man has a partner, then the sequential number (or id) of his partner is shown. If a man is single, then we can mark his own sequential number as his partner.

Another way of coding is to also code in the constraints of the problem, but the constraints can be built into and evaluated by the fitness function. With constraints in the genetic code, the crossover function would be much more difficult to implement. That is why we opt for the simple coding.

For crossover one has to take two solutions and swap some part(s) of the code (as following):

1. Find the men that have different partners.

2. Randomly select some of those men and swap their partners between two solutions.
3. Among the pairs that weren't changed find the women that have more than one partner in one solution. Break up these pairs.
4. If unstable pairs are not allowed, then find unstable pairs and mark them.
5. Search for a good solution (using a non-GA algorithm) among marked pairs and single men and women. Don't search too far.

The mutation algorithm is following:

1. Take one solution and find the men who have partners.
2. Randomly select some of those men and shuffle their partners.
3. If necessary, repeat step 2.
4. If unstable pairs are not allowed, then find unstable pairs and mark them.
5. Search for a good solution (using a non-GA algorithm) among marked pairs and single men and women. Don't search too far.

A fitness function is used to select the best solutions. It is wise to construct a relative fitness function that compares two solutions, instead of an absolute fitness function. Majority voting, for example, is feasible only as a part of a relative fitness function. If an absolute fitness function is applicable, then one can always use n-1 relative comparisons to get absolute fitness values for n solutions.

There are several ways how to find the best solutions from the population. A simple approach to tournament ranking is called tournament selection. In tournament selection, some number of individuals are randomly chosen from the population, the best from the chosen individuals are copied to the intermediate population (Blickle et al., 1995). This is repeated until the selection number is reached. Often tournaments are only held between two individuals (called binary tournaments).

A major problem with comparing solutions is the transitivity of ranking. Transitivity requires that if solution a is better than solution b and solution b is better than solution c, then solution a must be better than solution c. With special constraints (for example permitting couples to submit combined preferences) this transitivity may not always hold. Tournament selection does not minimize transitivity inconsistencies. In this situation it is necessary to use a full tournament to get a correct ranking (of solutions). The minimal transitivity inconsistency ranking in a tournament problem is NP-hard, preventing the use of optimal ranking methods on tournament tables bigger than 20-30 objects (for our framework that number would have to be even smaller due to time constraints). That is why an efficient polynomial-time heuristic tournament method is needed, that can cope with thousands of objects. Ali, Cook and Kress have conducted a study where they compared two heuristic tournament algorithms, namely Iterated Kendall method (IK) (Ali et al., 1986) that is derived from a Kendall method (Kendall, 1962) and p-connectivity method by Goddard (Goddard, 1983). Our proposed tournament method for the framework is based on a class of heuristic methods that make use of monotone systems (Mullat, 1976; Võhandu, 1989, 1990).

The following chapter describes a matching market where the set of stable matchings constituting the core may be empty and there is a need to relax the definition of the core. In doing so the matchings in the new core are not transitive and it is necessary to use tournament ranking to pick out the best matching.

3. AN EXTENDED CORE OF A MARRIAGE MODEL WITH INTRANSITIVITIES

In the one-to-many matching model with couples, the set of stable matchings and consequently the core of the marriage game may be empty, as shown by Roth with an example (Roth et al., 1990, theorem 5.11, page 141). If we study the dominance between the 24 unstable matchings of that example, it becomes clear that every dominance path leads to the following cycle of unstable matchings {6, 20, 19, 5}. From that we can deduce the following theorem.

Theorem 1. If the set of stable matchings is empty, but the set of unstable matchings is not empty, then every arbitrary dominance path leads to some cycle of unstable matchings.

For proof it is sufficient to say that the set of unstable matchings is finite and therefore every dominance path either leads to some stable matching (a contradiction) or to some cycle of unstable matchings.

It is not known (to us) whether there can be only one such cycle of unstable matchings or more. If there is only one such cycle, then we can say that the matchings in that cycle dominate over all other unstable matchings not in the cycle. And the matchings within the cycle dominate over each other. So it can be argued, that two-way domination is canceled out and it is not actually a domination at all. This is a relaxation of the weak dominance, that we call cyclic domination.

Definition 1. For any two feasible game outcomes of x and y, x cyclically dominates y if and only if there exists a coalition of players S such that

(a) every member of the coalition S prefers x at least as much as y; and
(b) at least one member of the coalition S prefers x to y; and
(c) the rules of the game give the coalition S the power to enforce x (over y); and
(d) y does not weakly dominate x through some arbitrary length sequence of weak dominations.

Definition 2. The core of a game defined by cyclic domination is the set of cyclically undominated outcomes.

The definitions (3 and 4) of a weak domination are taken from Roth (Roth et al., 1990; pages 54-55, 166-167) and are as follows:

Definition 3. For any two feasible game outcomes x and y, x weakly dominates y if and only if there exists a coalition of players S such that

(a) every member of the coalition S prefers x at least as much as y; and
(b) at least one member of the coalition S prefers x to y; and
(c) the rules of the game give the coalition S the power to enforce x (over y).

Definition 4. The core of a game defined by weak domination is the set of weakly undominated outcomes.

The existence of cyclic domination also means the existence of intransitivity. So to reach an outcome, the players have to vote between pairwise matchings as in a tournament.

Since the length of the cycle of unstable matchings and the length of the domination path leading to it is unknown, the problem of finding the cycle could still be hard. And if there are more than one cycle, then finding the core becomes even more difficult. Our proposed framework should suite well to find the matchings constituting the core.

The following chapter describes a tournament between the matchings in the newly defined core. The tournament can be viewed apart from or as part of the framework. If the framework does indeed find the core, then at the final iteration tournament ranking gives the best matching from the core matchings. The proposed tournament algorithm is based on monotone systems.

4. EXAMPLE TOURNAMENT BASED ON MONOTONE SYSTEMS

Definition 3. (A weakly) monotone system is a system built on a set of objects, such that

(a) objects are weighted by a value function
(b) after removal of one object from the set all the weights of other objects still in the set change monotonically in one direction (increase or decrease) or stay on the same level.

Algorithms based on such a simple monotone system work as follows:

Step 1. Evaluate all objects in the set.

Step 2. Find the weakest object (with the smallest (largest) weight), and remove it from the set. If there are several weakest objects, then recursively apply the tournament algorithm to the set of weakest objects. If at any stage of the recursion any object was removed from the set of weakest objects, then backtrack. If the set of weakest objects still contains more than one object, then compare the weights from the previous iteration and choose an object that is more similar to the previously removed object. If the weights in all the previous iterations are the same, then according to the value function these objects are equivalent and we can remove any one of those (usually the first object will be removed).

Step 3. If there are still objects in the set, then continue from Step 1.

Any given algorithm always removes the object with the smallest weight, or the largest weight. Algorithm cannot change the choice function (min, max) during the course of action. Value function can be chosen relatively freely, as long as it satisfies monotonicity condition. The sequence of removal of objects constitutes object ranking.

In a majority voting, all the players have to pairwise vote between the matchings. The tournament table is computed based on the voting table. We compare votes of all pairs of matchings v_{rc} and v_{cr} and make the following transformations:

If $v_{rc} < v_{cr}$ then $t_{rc} = 0$, $t_{cr} = 1$;
If $v_{rc} > v_{cr}$ then $t_{rc} = 1$, $t_{cr} = 0$;
If $v_{rc} = v_{cr}$ then $t_{rc} = 0$, $t_{cr} = 0$.

The process of finding the weakest object to remove is iterative – the weakest objects are selected by the minimum number of wins and then by the maximum number of losses in the remaining subset of the weakest objects. This iterative minimax selection is used until either only one weakest object remains or the last minimax selection was not able to reduce the number of weakest objects. In the latter case the first remaining weakest object in the original ranking is removed. The algorithm is as follows:

1. Find rowsums, column sums.
2. Set of weakest objects = set of objects remaining in the tournament table
3. Select from the set of weakest objects the objects with the smallest rowsums (the least number of wins).
4. If there was a selection, then mark the occurrence of it. Recompute selection table rowsums and column sums. If a single weakest object is remaining, then resume from step 8.
5. Select from the set of weakest objects the objects with the largest column sums (the biggest number of losses).
6. If there was a selection, then mark the occurrence of it. Recompute selection table rowsums and column sums.
7. If the set of weakest objects contains more than one object and the selection has occurred during steps 3 and/or 5, then resume from step 3.
8. Add the first remaining weakest object to the top of the ranking list and remove it from the tournament table, recompute tournament table rowsums and column sums.
9. If the tournament table still contains objects, then resume from step 2.
10. The end.

The finding and removal of the weakest object has a maximum complexity of $O(N^2)$, because each remaining object in the set of weakest objects can be removed only once. The removal of an object from either tournament table or from the set of weakest objects has a time complexity of $O(N)$, because the row of the removed object has to be negated from rowsums and the corresponding column has to be negated from column sums. So, the algorithm has a maximum time complexity $O(N^3)$.

The tournament results for the matchings in the cycle {5, 6, 19, 20} are in Table 1.

Table 1. Tournament table with wins and losses in subsequent iterations

t	5	6	19	20	Iter1	Iter2	Iter3	Iter4
5		0	1	1	2	1	0	0
6	0		0	0	0	0	0	
19	0	0		1	1	0		
20	0	0	0		0			Wins
Iter1	0	0	1	2				
Iter2	0	0	1					
Iter3	0	0						
Iter4	0			Losses				

In the first iteration matchings 6 and 20 have no wins, but the number of losses are 0 and 2 accordingly. Matching 20 is removed first based on the number of losses. Values from column 20 are subtracted from the winning points (row sums) of remaining matchings. Values from row 20 are subtracted from the losses (column sums) of remaining matchings.

In the second iteration matchings 6 and 19 have no wins. Based on the number of losses (0 and 1) matching 19 will be removed. Wins and losses of the remaining matchings are recalculated.

In the third iteration matchings 5 and 6 have no wins. Voting between them gave a draw. Both have no losses, since voting between them gave a draw. One way to differentiate between the two matchings is to look at the wins (and then losses) before the first iterations. Matching 5 had one win in the previous iteration, so matching 6 has to be removed first and matching 5 will be removed last.

The proposed method has been tested to give a ranking with minimum number of transitivity faults on all tournament tables (including ties) up to size 5x5 (detailed results of the experiment are in a paper submitted to a conference CAISE'04).

5. CONCLUSION

A matching framework has been proposed that always finds a solution for NP-hard and APX-hard matching problems. It is based on a genetic algorithm that uses intermediate or approximate solutions from other matching algorithms. The best matchings from each population are chosen by a tournament ranking.

We have also proposed a tournament algorithm based on monotone systems and a value function for it. The proposed algorithm has a maximum time complexity $O(N^3)$ and it has been tested to give a ranking with minimum number of transitivity faults on all tournament tables (including ties) up to size 5x5.

We have described a matching model, where intransitivities may arise and show how the proposed tournament method works.

We have also defined a more relaxed core definition for the marriage models where the set of stable matchings may be empty.

ACKNOWLEDGEMENT

This work was partially supported by ESF Grant 4844.

REFERENCES

Ali, I., Cook, W.D., Kress, M., On the Minimum Violations Ranking of a Tournament, *In Management Science*, Vol.32, No.6, June 1986, pp. 660-672.

Blickle, T., Thiele, L., 1995. A Mathematical Analysis of Tournament Selection, *In Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 9-16.

Gale, D., Shapley, L., 1962. College Admissions and the Stability of Marriage. *In American Mathematical Monthly*, 69(1), pp.9-15.

Gärdenfors, P., 1975. Match Making: Assignments Based on Bilateral Preferences, *In Behavioral Science*, 20, pp.166-173.

Goddard, S.T., Ranking in Tournaments and Group Decisionmaking, *In Management Science*, Vol.29, No.12, Dec. 1983, pp.1384-1392.

Gusfield, D., Irwing, R. W., 1989. *Stable Marriage Problem: Structure and Algorithms*, MIT Press Series in the Foundations of Computing, MIT Press, Cambridge, Massachusetts, USA.

Halldórsson, M., Iwama, K., Miyazaki, S., Morita, Y., 2002. Inapproximability results on stable marriage problems, *In Proceedings of LATIN 2002: the Latin-American Theoretical INformatics symposium*, volume 2286 of Lecture Notes in Computer Science, pp. 554-568. Springer-Verlag.

Iwama, K., Manlove, D., Miyazaki, S., Morita, Y., 1999. Stable marriage with incomplete lists and ties, *In Proc. ICALP'99*, pp.443-452.

Kendall, M., *Rank Correlation Methods*, 3rd ed., Hafner, New York, 1962.

Klijn,F., Masso,J., 2003. Weak stability and a bargaining set for the marriage model, *In Games and Economic Behavior*, 42, pp. 91-100.

Mullat, I., 1976. Extremal Subsystem of Monotone Systems. *In Automation and Remote Control*, 5, pp. 130-139; 8, pp. 169-178 (in Russian), http://www.datalaundering.com/download/extrem01.pdf.

Roth, A., Sotomayor, M., 1990. *Two-sided matching : a study in game-theoretic modeling and analysis*, Cambridge University Press, Econometric Society Monographs, no.18, Cambridge, Massachusetts, USA.

Võhandu, L., 1989. Fast Methods in Exploratory Data Analysis, *In Proceedings of Tallinn Technical University*, Tallinn, No. 705, pp. 3-13, http//www.datalaundering.com/download/fastmeth.pdf.

Võhandu, L., 1990. Best Orderings in Tournaments, *In Proceedings of Tallinn Technical University*, Tallinn, No. 723, pp. 8-14.