Solving Stable Matching Problems via Cooperative Parallel Local Search

Danny Munera

Abstract: Stable matching problems and its variants have several practical applications, like the Hospital/Residents problem, stable roommates problem or bipartite market sharing. An important generalization problem is the SMTI which allows for incompleteness and ties in the user's preference lists. Finding a maximal size stable matching for SMTI is computationally difficult. We developed a Local Search method to solve SMTI using the Adaptive Search algorithm and present experimental evidence that this approach is much more efficient than state-of-the-art exact and approximate methods (in terms of both computational effort required and quality of solution). We also tried a parallel version of our algorithm. For this we reused the Cooperative Parallel Local Search framework (CPLS) we designed. CPLS is a highly parametric framework for the execution in parallel of local search solvers allowing them to cooperate though communication. The cooperative parallel version of our local search algorithm improves performance so much that very large and hard instances can be solved quickly.

Solving Stable Matching Problems via Cooperative Parallel Local Search

Danny Munera

University Paris 1 (Pantheon-Sorbonne), France
Danny.Munera@malix.univ-paris1.fr

Keywords: *stable matching, local search, adaptive search, parallelism*

1 Introduction

Stable Marriage (SM) problem [2] is the problem to find a *stable matching* between a set of n men and a set of n women, each of whom have ranked all members of the other set in a strict order of preference. This match is stable if there is no man-woman pair where both would rather marry each other than their current partner - such a pair is called a *blocking pair*. A generalization of this problem is the Stable Marriage with Incomplete List and Ties (SMTI) [7]. In SMTI, the preference lists may include ties (to express indifference) and may be incomplete (to express that some partners are unacceptable). The goal now is to find the stable matching of maximal size (that is, with the smallest number of singles). This problem is NP-hard [6].

Stable matching problems have many practical applications such as car-sharing or bipartite market sharing, job markets and social networks. Another important application is the Hospitals/Residents problem (HR) [7] which consists in assigning resident doctors to hospitals, based on their preferences. For HR, there are nation-wide programmes in various countries like the Scottish Foundation Allocation Scheme (SFAS), the Canadian Resident Matching Service (CARMS) or the the National Resident Matching Program (NRMP) in the USA. Many of these applications involve very large sets and it is thus a real challenge to design efficient algorithms (in terms of *execution time* and/or *solution quality*).

We address the SMTI problem using Local Search (LS) and parallelism. For this, we first proposed a Cooperative Parallel Local Search framework (CPLS) [9] which allows the user to run several Local Search solvers in parallel with sophisticated forms of cooperation (through communication). This framework is highly parametric and can be customized for different situations (e.g. the user can modify the LS algorithm to use, the level of "intensification" and "diversification" in the search, the communication topology). Due to space limitation, we do not describe more the CPLS framework, the interested reader may refer to [9].

We then proposed a Local Search algorithm based on the Adaptive Search (AS) method [10]. For this, we showed how to model SMTI problems as permutation problems to limit the neighbourhood to explore. Furthermore, we designed efficient heuristics based on the properties of the blocking pairs to improve the current assignment. Other heuristics are also proposed to escape local minima. The resulting algorithm (called AS-SMTI) can be implemented efficiently with a compact data representation. We present experimental evidence that this sequential algorithm is much more efficient than state-of-the-art exact and approximate methods. Moreover, we developed a parallel implementation of the AS-SMTI using the CPLS framework (written in the X10 language [1]). We show that the cooperative parallel version exhibits super-linear speedup on average and behaves particularly well on hard instances.

2 A Local Search solver for SMTI

In AS-SMTI, we model SMTI as a permutation problem : the sequence of n $(X_1 \dots X_n)$ takes on as values permutations of the values $1 \dots n$ (implementing an `all-different` constraint). $X_i = j$ is interpreted as either (m_i, w_j) in a *match* M, or m_i is single if w_j is not on its preference list. The AS-SMTI algorithm starts from a random matching, then it explores a limited neighbourhood based on a heuristic which selects the "worst" blocking-pair (BP) to fix and/or a single man to marry. If a pair (m, w) forms a BP in M, the error associated to this BP is the distance between the woman w and the current partner of m ($X_m = w'$) in the preference list of m. Thus, the further the assigned woman w' is from w, the larger the error. To fix the "worst" BP, AS-SMTI swaps the values of the BP variable $X_m = w'$ and the variable that contains the desired match, $X_i = w$ (i.e. two men exchange their partner).

The total cost function of a matching M measures both its stability (number of BPs) and its quality (number of singles). Hence : $cost(M) = \#BP(M) \times n + \#Singles(M)$ where $\#BP(M)$ is the number of BPs in M, and $\#Singles(M)$ is the number of singles in M. The number of BPs is weighted with n to prioritize stable marriages over marriages with fewer singles.

Finally if a local minimum is reached, AS-SMTI executes a customized reset procedure which basically tries to fix the 2 worst BPs and/or to assign a woman to a single man. This procedure is stochastic ; it will *also* fix the second worst variable with a probability p : good results are obtained with a high value, e.g. $p \simeq 0.98$.

The algorithm stops when a *perfect solution* (PS) is found (a stable matching with no singles) or when a given timeout is reached (the best matching found so far is then returned).

2.1 Performance Evaluation

We compared our AS-SMTI algorithm to other approaches both from the point of view of *solution quality* and of pure *performance*. For this comparison we selected three different methods : another Local Search method (the LTIU algorithm of [3]), a very efficient 3/2-approximation algorithm (the McDermid's method [8]) and a SAT approach (using the encoding proposed in [5]). For this evaluation, we used the random problem generator described in [4] which takes three parameters : the size (n), the probability of incompleteness ($p1$) and the probability of ties ($p2$). We generated problems of size $n = 100$, with $p1$ ranging over $[0.1, 0.9]$ and $p2$ over $[0, 1]$, with step 0.1. We used an X10 [1] implementation of AS-SMTI, running sequentially on an AMD Opteron 6376 clocked at 2.3 GHz, i.e. using only one core.

Regarding *solution quality*, we measured the percentage of perfect solutions (PS) reached by all algorithms. In most of the cases, the solutions returned by AS-SMTI are better than solutions returned by its opponents. On average, AS-SMTI is slightly better than LTIU. However, LTIU reports a dramatic lost of performance when $p2 = 1$ obtaining less than 60% of PS while AS-SMTI reaches 100%. For McDermid (MD), the percentage of PS found by AS-SMTI is considerably higher than this obtained by MD, in particular using a probability of ties $p2 \in [0.1..0.7]$. The reference SAT encoding was restricted to the decision problem : *is there a stable matching of size n ?* which we answer by actually finding a perfect stable matching. AS-SMTI always found a PS for the tested cases.

Regarding the *performance* (which measures the computational effort), Figure 1 compares AS-SMTI with its opponents. AS-SMTI is much faster that LTIU and SAT (thus the log Y scale) : the LTIU method averages over $30\,s$ on a similar machine while AS-SMTI is two orders of magnitude faster (when $p2$ increases, the lead extends even further). Similarly, AS-SMTI outperforms the SAT version by a factor of about 50. From the raw performance point of view, the comparison with McDermid is particularly significant. Recall that an approximation algorithm gives priority to the speed (running in linear time) at the expense of the quality. Surprisingly, in many cases, AS-SMTI is up to an order of magnitude faster than MD. However, when the probability of incompleteness is very high (e.g. $p1 = 0.9$) MD outperforms AS-SMTI. It is worth noticing that MD always returns the same, single and (sub)optimal solution, while AS-SMTI will yield more than one solution, with observably better quality. Moreover,

a *solution quality* vs. *performance* trade-off is always possible in AS-SMTI, by tweaking the timeout parameter.

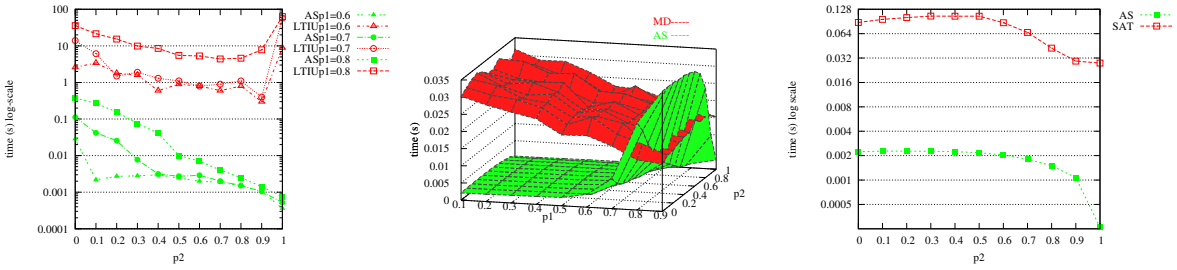

FIG. 1 – Execution time comparing AS-SMTI to LTIU, MD and SAT (times in s)

3 Parallel Evaluation

We also experimented with a parallel version of AS-SMTI. For this, we used the Cooperative Parallel Local Search (CPLS) framework we previously proposed [9]. Due to space limitation we do not detail how we instantiated and tuned CPLS for the SMTI case (see [10]). We attacked problems of size 1000 generated with $p1 = 0.95$ and $p2 = 0.8$. These problems involve a large number of variables, a huge search space ($1000! \simeq 10^{2567}$) and are difficult to solve due to the high level of incompleteness in the preference lists. All selected problem admit a perfect solution. We generated two different sets of problems. The first set, called *normal*, is composed by 10 random SMTI problems of size $n = 1000$. The second set, called *hard*, was generated from 100 random problems of size $n = 1000$, we ran them sequentially and selected the 10 hardest instances.

Each problem was executed 50 times (the results are averaged), varying the number of cores from 1 (sequential) to 128 and using an unlimited timeout to force the solver to discover the perfect solution. The used architecture is a cluster of 4 machines, each with 4×16-core AMD Opteron 6376 CPUs running at 2.3 GHz and 128 GB of RAM.

Figure 2 presents a log-log graph of execution times. The *Ideal time* corresponds to a linear speedup : time is halved when the number of cores is doubled. On the *normal* data set, our parallel AS-SMTI is rather efficient : using 80 cores the average time decreases from $44.5s$ to $0.519s$ which corresponds to a speedup factor of 86. Beyond this number of cores, the time tends to stabilize. This could mean that we have reached an incompressible limit to solve problems of size 1000, or that we are being limited by low-level factors.

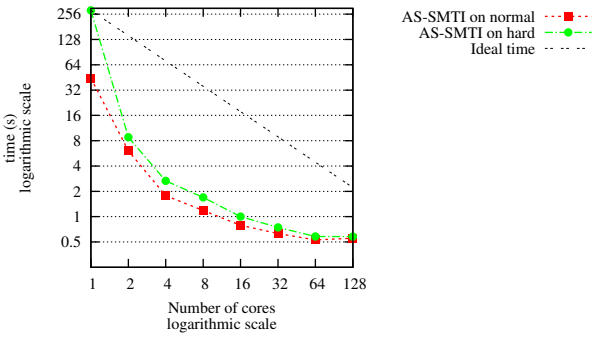

FIG. 2 – Execution time for *normal* and *hard* problems varying the number of cores.

Regarding the *hard* problems, the difficulty of the instances is clear from the sequential time : in the normal experiment, a problem was solved in about $44s$ while about $285s$ are needed for the hardest instances. Using the cooperative search, execution times are drastically improved and the best speedup is 492 with 128 cores which corresponds to a reduction of the execution time from $284.5s$ to $0.579s$. It is worth noticing that this time is very similar to the best time ($0.519s$) obtained on the normal set. From a practical point of view, it appears that parallelism with cooperation neutralizes the relative difficulty of problems, as instances which are originally about 6 times harder get solved in approximately the same time. Of course, the problem retains its worst-case NP-hard complexity, and parallel search cannot change this.

4 Conclusion and Work in progress

We proposed the AS-SMTI algorithm which models SMTIs as permutation problems and solves them using a local search approach based on Adaptive Search. AS-SMTI outperforms state-of-the-art solvers for SMTIs. Moreover, the cooperative parallel version shows super-linear speedup and performs exceptionally well, particularly on very hard instances. We plan to experiment with very large instances on a massively parallel machine to test the scaling limitations. We also plan to study which features of the stable matching problem make it so suitable for cooperation.

Currently we are attacking the Hospital/Resident problem with Ties (HRT). We have already developed a solution based on model transformation : an HRT problem is transformed into SMTI which is handled by our AS-SMTI solver (slightly modified). The preliminary results on a sequential machine are very encouraging : execution times are much faster than the best Integer Programming approach while obtained solution are very close to the optimum [1]. We are now experimenting with parallelism on HRT. We also plan to improve the core solver to optimize features appearing when transforming HRT to SMTI (such optimizations will also benefit to other SMTI problems).

Acknowledgments

I would like to thank all the co-authors of the papers included on this work for their contributions, help, and encouragements throughout this project.

Références

[1] P. Charles, C. Grothoff, V. Saraswat, C. Donawa, A. Kielstra, K. Ebcioglu, C. von Praun, and V. Sarkar. X10 : an object-oriented approach to non-uniform cluster computing. In *SIGPLAN*, pages 519–538, San Diego, CA, USA, 2005. ACM.

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

[3] M. Gelain, M. Pini, F. Rossi, K. Venable, and T. Walsh. Local Search Approaches in Stable Matching Problems. *Algorithms*, 6(4) :591–617, Oct. 2013.

[4] I. P. Gent and P. Prosser. An Empirical Study of the Stable Marriage Problem with Ties and Incomplete Lists. In *in ECAI 2002*, pages 141–145. IOS Press, 2002.

[5] I. P. Gent, P. Prosser, B. M. Smith, and T. Walsh. SAT encodings of the stable marriage problem with ties and incomplete lists. In *SAT*, volume 8, pages 133–140, Cincinnati, USA, 2002.

[6] K. Iwama, D. Manlove, S. Miyazaki, and Y. Morita. Stable Marriage with Incomplete Lists and Ties. In *ICALP'99*, pages 443–452. Springer-Verlag, 1999.

[7] D. F. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, and Y. Morita. Hard variants of stable marriage. *Theoretical Computer Science*, 276(1-2) :261–279, 2002.

[8] E. J. McDermid. A 3/2 -approximation algorithm for general stable marriage. In *ICALP'2009*, pages 689–700, Rhodes, Greece, 2009.

[9] D. Munera, D. Diaz, S. Abreu, and P. Codognet. A Parametric Framework for Cooperative Parallel Local Search. In *EvoCOP'2014*, Granada, Spain, 2014.

[10] D. Munera, D. Diaz, S. Abreu, F. Rossi, V. Saraswat, and P. Codognet. Solving Hard Stable Matching Problems via Local Search and Cooperative Parallelization. In *AAAI*, 2015.

1. We have submitted an article to the MATCHUP 2015 (a workshop dedicated to stable matching).