Distributed Stable Matching Problems with Ties and Incomplete Lists

Ismel Brito and Pedro Meseguer

Institut d'Investigació en Intel.ligència Artificial
Consejo Superior de Investigaciones Científicas
Campus UAB, 08193 Bellaterra, Spain.
{ismel|pedro}@iiia.csic.es

Abstract. We consider the Stable Marriage Problem and the Stable Roommates Problem, two types of the general class of Stable Matching Problems, in presence of ties and incomplete preference lists. They can be solved by centralized algorithms, but this requires to make public the preference lists of members, something that members would prefer to avoid for privacy reasons. This motivates a distributed formulation of these problems to keep privacy. We propose a distributed constraint approach that solves all the considered problems, keeping privacy.

1 The Stable Marriage Problem

The Stable Marriage Problem (SM) [5] involves n men and n women. Each man m_i ranks women forming his preference list, and each woman w_j ranks men forming hers. A matching M is a one-to-one mapping between the two sexes. M is *stable* when there is no blocking pair (m, w) such that m and w, no partners in M, prefer one another to his/her partner in M. A *solution* is a stable matching.

There are several SM versions. In the Stable Marriage with Incomplete Lists (SMI) some people may consider unacceptable some members of the other sex. In the Stable Marriage with Ties (SMT), some people may consider equally acceptable some members of the other sex, so there is a tie among them. For SMT, three stability types have studied: weak, strong and super [6]. (m, w) is a weak blocking pair for M if m and w are not partners in M, and each of whom strictly prefers the other to his/her partner in M. (m, w) is a strong blocking pair for M if m and w are not partners in M, and one strictly prefers the other to his/her partner in M and the other is at least indifferent between them. (m, w) is a super blocking pair for M if m and w are not partners in M, and each of whom either strictly prefers the other to his/her partner in M or it is indifferent between them. In the Stable Marriage with Ties and Incomplete Lists ($SMTI$), some person may consider as equally acceptable some members of the other sex, while others are unacceptable. The three stability types also apply here.

Solvability conditions, complexity and solving algorithms of each SM version appear in Table 1. For *SMTI-weak*, different solutions may exist with different lengths, so it is of interest to find the matching of maximum cardinality. This is *SMTI-weak-max*, an optimization problem that is NP-hard.

SM version	$\exists$ solution?	Size	All solutions		Algorithm	Compl
			Length	Partners		
SM	always	n	same	same	*EGS* [5]	poly
SMI	always	$\leq n$	same	same	*EGS* [5]	poly
SMT-weak	always	n	same	same	break ties + *EGS* [6]	poly
SMT-strong	not always	n	same	same	*STRONG* [6]	poly
SMT-super	not always	n	same	same	*SUPER* [6]	poly
SMTI-weak	always	$\leq n$	diff	diff	break ties + *EGS* [7]	poly
SMTI-strong	not always	$\leq n$	same	same	*STRONG2* [7]	poly
SMTI-super	not always	$\leq n$	same	same	*SUPER2* [7]	poly
SMTI-weak-max	always	$\leq n$	same	diff	break ties in all possible ways + *EGS* [7]	NP-hard

Table 1. Solvability conditions, solving algorithm and complexity for the *SM* versions.

$DisSM$ problem	Centralized Algorithm	Extension to the distributed case, keeping privacy
DisSMT-weak	break ties + *EGS* [6]	break ties arbitrary + *DisEGS* [3]
DisSMT-strong	*STRONG* [6]	No extension [2]
DisSMT-super	*SUPER* [6]	Extension [2]
DisSMTI-weak	break ties + *EGS* [7]	break ties arbitrary + *DisEGS* [3]
DisSMTI-strong	*STRONG2* [7]	No extension [2]
DisSMTI-super	*SUPER2* [7]	No extension [2]
DisSMTI-weak-max	break ties in all possible ways + *EGS* [7]	Discussion [2]

Table 2. Algorithms for solving *SM* and *DisSM* versions.

SM appears to be naturally distributed. Each person would like to keep his/her preference lists private, which is not possible in the centralized case. This motivates the Distributed Stable Marriage (*DisSM*) [3], defined as *SM* plus a set of $2n$ agents. Each agent owns exactly one person. An agent knows all the information of its owned person, but it cannot access the information of people owned by other agents. A solution is a stable matching. Similarly, we define here the distributed versions with incomplete lists (*DisSMI*), with ties (*DisSMT*) and with ties and incomplete lists (*DisSMTI*) [1].

Is it possible to extend the centralized algorithms to the distributed setting keeping privacy? *DisEGS* is a distributed version of Extended Gale-Shapley (*EGS*) that maintains privacy. It was used to solve *DisSM* and *DisSMI* [3]. Here we focus on *DisSMT* and *DisSMTI*, that jointly with the three stability types, produce six decision problems plus one optimization problem. The extension of centralized algorithms to the distributed case appear in Table 2. Only three out of the six decision problems can be solved by extending the centralized algorithms to the distributed case while keeping preferences private. Details appear in [2].

2 Constraint Formulation

In [4], *SM* is modeled and solved as a binary CSP with $2n$ variables. Variable domains are the preference lists. Constraints are defined between men and women: C_{ij} is a table with all possible partial matchings involving man i and woman j. For any pair k, l ($k \in Dom(i)$, $l \in Dom(j)$), the element $C_{ij}[k,l]$ represents the partial matching $(m_i, w_k)(m_l, w_j)$; which could be: **A**llowed, **I**llegal, **B**locked or **S**upport. Tables with A, I, B, S are passed into 1/0 by A, S $\rightarrow$ 1, I, B $\rightarrow$ 0.

[1] [9] proposes a solving method for all these versions based on encryption techniques.

For instances with ties, we consider the different definitions of stability. This affects the usage of **B**locked pair in C_{ij}. The definition given in [4] is valid for weak stability. For strong and super stability, we replace **B** definition in [4] by,

- $C_{ij}[k,l]$ = **B**locked if (m_i, w_j) is a strong blocking pair for (m_l, w_k).
- $C_{ij}[k,l]$ = **B**locked if (m_i, w_j) is a super blocking pair for (m_l, w_k).

Privacy is one of the main motivations for distributed CSP ($DisCSP$). We differentiate between value privacy and constraint privacy [1]. Value privacy implies that agents are not aware of other agent's values during the solving process and in the final solution. On constraint privacy, the Partially Known Constraints (PKC) model was presented. It assumes that when two agents i, j share a constraint C_{ij}, none of them fully knows it. Each agent knows the part of the constraint that it is able to build, using its own information. We say that agent i knows $C_{i(j)}$, and j knows $C_{(i)j}$. Similarly to [3], we use the constraint formulation of Section 2 to solve $DisSMT$ and $DisSMTI$ by the $DisFC$ algorithm [1], keeping privacy of preference lists using the PKC model.

How can i build $C_{i(j)}$? Ordering rows of $C_{i(j)}$ following his preference list, all elements in rows above w_j are 1 (except m_i^{th} column that are 0). All elements in rows below w_j may be 1 or 0, depending on the ordering of columns (except m_i^{th} column that are 0). Since x_i does not know the preference list of y_j, columns are ordered lexicographically, and the elements below w_j row are ? (undecided). If there is a tie between w_j and $w_{j'}$ elements in $w_{j'}$ row are # (tie). $C_{i(j)}$ is,

$$C_{i(j)} = \begin{array}{c|ccccccc} & m_1 & \dots & & m_i & & \dots & m_n \\ w_{i_1} & 1 & \dots & 1 & 0 & 1 & \dots & 1 \\ & \dots & & & & & & \\ & 1 & \dots & 1 & 0 & 1 & \dots & 1 \\ w_j & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ w_{j'} & \# & \dots & \# & 0 & \# & \dots & \# \\ & ? & \dots & ? & 0 & ? & \dots & ? \\ & \dots & & & & & & \\ w_{i_n} & ? & \dots & ? & 0 & ? & \dots & ? \end{array}$$

A property of these tables is that all columns of $C_{i(j)}$ are equal, except m_i column. All rows of $C_{(i)j}$ are equal except w_j row [3]. $C_{ij} = C_{i(j)} \diamond C_{(i)j}$, $\diamond$ operates component to component. $\diamond$ depends on each type of stability.

How does a distributed algorithm achieve a global solution?. $DisFC$ does it, using phase I only. $DisFC$ instead of sending the assigned value to lower priority agents, it sends the domain subset that is compatible with the assigned value. Also, it replaces actual values by sequence numbers. After assignment, each man agent sends the domain that each woman may take to the women involved. For example, when an agent i assigns value k to x_i, it sends to j the row of $C_{i(j)}$ corresponding to value k. This row may contain 1's, 0's, ?'s or #'s. For each received domain, the agents search for a compatible value. If the domain contains ? or # entries, they are disambiguated, following the rules of $\diamond$ operation specific for each type of stability. When j receives a domain with ? or # values, it performs the $\diamond$ operation with a row of $C_{(i)j}$ different from i. Since all rows in $C_{(i)j}$ are equal, except i row, they will all give the same result. The $\diamond$ operation j will compute the corresponding row in the complete constraint C_{ij}, although j does not know to which value this row corresponds. After this operation the

resulting domain will contain neither ? nor # values, and the receiver will have no ambiguity. If it finds no compatible value, it performs backtracking. The process repeats until finding a solution or detecting that no solution exits.

The $\diamond$ operation depends on the stability type. *Weak Stability.* From *weak blocking pair* definition, we realize that no matched pair (m, w) will be blocked for any other pair (m',w'), such that either m is indifferent between w and w' or w is indifferent between m and m'. So #'s are replaced by 1's in tables. Rules for the $\diamond$ operation of [3] apply. For *DisSMTI-weak*, not all stable matchings have the same length. One may desire to find a matching of maximum cardinality. With this aim, we consider the question 'Is there a *weakly stable* matching of size k?', where k starts with value n. If a weakly stable matching exits, it will be of maximum cardinality. Otherwise, the value k is decreased by one, and the problem is reconsidered. Modeling this idea with constraints, we add n variables, $u_1, u_2, \ldots, u_n$, plus an extra variable z, with the domains: $D(u_i) = \{0, 1\}, 1 \leq i \leq n$, $D(z) = \{k\}$. New constraints are: if $x_i < n + 1$ then $u_i = 1$ else $u_i = 0, 1 \leq i \leq n$ and $z = \sum_{i=1}^{n} u_i$. The agent that owns x_i also owns u_i. An extra agent owns z. *Strong Stability.* $\diamond$ includes #: $\# \diamond \# = 1$, $1 \diamond \# = 1$, $\#\diamond? = 0$, $0 \diamond \# = 0$. *Super Stability.* $\diamond$ has a single change from strong stability:$\# \diamond \# = 0$.

Experimentally, we observe that when it is possible to extend an specialized algorithm to the distributed setting while keeping preference private, the resulting algorithm is more efficient than $DisFC$. However, only a fraction of the SM versions can be solved by the specialized distributed algorithms, while all of them can be solved by the generic distributed constraint formulation.

3 The Stable Roommates Problem

The Stable Roommates Problem (SR) consists of $2n$ participants, each ranks *all* other participants in strict order according to his/her preferences [5]. A matching is a set of n disjointed participant pairs. A matching is *stable* if it contains no *blocking pair*. A pair (p_i, p_j) is a blocking pair for M if p_i prefers p_j to his/her partner in M and p_j prefers p_i to his/her partner in M. There are instances that admit no stable matching. The goal is to determine whether an SR instance is solvable, and if so, find a stable matching. Like SM, there are versions with Incomplete Lists (SRI), with Ties (SRT), and with Ties and Incomplete Lists ($SRTI$). For versions including Ties, there are three stability types, weak, strong and super. Solvability conditions, complexity and solving algorithms of each SR version appear in Table 3. Considering *SRTI-weak*, different solutions may exist with different lengths, so it is of interest to find the matching of maximum cardinality. This is *SRTI-weak-max*, an optimization problem that is NP-hard.

With a motivation similar to SM, the Distributed Stable Roommates Problem ($DisSR$) [3] is defined by $2n$ persons plus a set of agents. Each person ranks all the other in his/her preference list. Each agent owns exactly one person. An agent knows the preference list of its owned person, but it does not know others' preferences. A solution is to find a stable matching, if it exists. Similarly, we de-

SR version	∃ solution?	Size	All solutions		Algorithm	Compl
			Length	Partners		
SR	not always	n	same	same	*Stable Roommates* [5]	poly
SRI	not always	$\leq n$	same	same	*Stable Roommates* [5]	poly
SRT-weak	not always	n	same	same	break ties in all possible ways + *Stable Roommates*, until finding a solution [8]	NP-comp
SRT-strong	not always	n	same	same	? [8]	?
SRT-super	not always	n	same	same	*SRT-super* [8]	poly
SRTI-weak	not always	$\leq n$	diff	diff	break ties in all possible ways + *Stable Roommates*, until finding a solution [8]	NP-comp
SRTI-strong	not always	$\leq n$	same	same	? [8]	?
SRTI-super	not always	$\leq n$	same	same	*SRTI-super* [8]	poly
SRTI-weak-max	not always	$\leq n$	same	diff	break ties in all possible ways + *Stable Roommates* [8]	NP-hard

Table 3. Solvability conditions, solving algorithm and complexity for the *SR* versions.

DisSR problem	Centralized Algorithm	Extension to the distributed case, keeping privacy
DisSRT-weak	break ties in all possible ways + *Stable Roommates*, until finding a solution [8]	No extension
DisSRT-strong	? [8]	No extension
DisSRT-super	*SRT-super* [8]	Extension
DisSRTI-weak	break ties in all possible ways + *Stable Roommates*, until finding a solution [8]	No extension
DisSRTI-strong	? [8]	No extension
DisSRTI-super	*SRTI-super* [8]	Extension
DisSRTI-weak-max	break ties in all possible ways + *Stable Roommates* [8]	No extension

Table 4. Algorithms for solving *SR* and *DisSR* versions.

fine the distributed versions with incomplete lists (*DisSRI*), with ties (*DisSRT*) and with ties and incomplete lists (*DisSRTI*).

We investigate if centralized algorithms can be extended to the distributed case keeping privacy, focusing on *DisSRT* and *DisSRTI*. Their resolution is summarized in Table 4. We conclude that only two decision problems can be solved by extending the centralized algorithms to the distributed case while keeping preferences private. Experimentally, we get similar results to those obtained *SM*.

References

1. Brito I., Meseguer P. Distributed Forward Checking. *Proc. CP-03*, 801–806, 2003.
2. Brito I., Meseguer P. Distributed Stable Marriage with Ties and Incomplete Lists. *Proc. ECAI-06 Workshop on Distributed Constraint Satisfaction*, 2006.
3. Brito I., Meseguer P. Distributed Stable Matching Problem.*Proc.CP-05*,152–166.
4. Gent I., Irving R. W., Manlove D. F., Prosser P., Smith B. M. A constraint programming approach to the stable marriage problem. *Proc. CP-01*, 225-239, 2001.
5. Gusfield D., Irving R. W. *The Stable Marriage Problem: Structure and Algorithms*. The MIT Press, 1989.
6. Irving R.W.Stable Marriage and Indifference.*Discr. Appl. Maths.*,48:261–272, 1994.
7. Manlove D.F. Stable marriage with Ties and Unacceptable Partners. *TR-1999-29*, Dep. Computing Science, Univ. Glasgow, 1999.
8. Irving, R. W., Manlove, D. F. The roommates problem with ties. *J. Algorithms*,43(1):85-105, 2002.
9. Silaghi, M.C. and Zanker,M. and Bartak,R., Desk-mates (Stable Matching) with Privacy of Preferences, and a new Distributed CSP Framework, *Proc. of CP'2004 Workshop on Immediate Applications of Constraint Programming*, 2004