Knowledge-Rich Stable Marriage Problems

Selin Eyupoglu, Berkan Teber, Ahmet Alkan, and Esra Erdem

Sabanci University, Istanbul, Turkey

Abstract. We study stable marriage problem and its optimization variants, augmented with further knowledge in addition to the rankings of individuals. The additional knowledge may be about specific preferences and constraints of individuals or groups of individuals, or general assumptions and social norms about them. We investigate the use of the knowledge representation and reasoning paradigm Answer Set Programming to solve such knowledge-rich variations of stable marriage problems.

Keywords: Matching · Stable marriages · Answer set programming

1 Introduction

Matching problems have been studied in economics, starting with the seminal paper of Gale and Shapley [8], which has led to a Nobel Prize in 2012, utilizing game theory methods with the goal of a mechanism design. Matching problems are about markets where individuals are matched with individuals, firms or items, typically across two sides, as in employment [22] (e.g., who works at which job), kidney donation (e.g., who receives which transplantable organ) [17, 21], and marriages [15, 8] (e.g., who marries with whom). In each problem, preferences of individuals, firms or items are given, possibly along with other information (e.g., the quotas of the universities in university entrance). The aim is to identify matchings that have nice properties such as stability, optimality or fairness.

While matching problems are defined mainly over preferences and quotas over resources [2, 1], real world applications may further require specific knowledge about some individuals or firms, general knowledge about their communities, traditions and social norms, and/or commonsense knowledge. For instance, depending on the matching problem, different constraints/preferences can be provided (e.g., some companies may want to hire more women, or some women may prefer to be matched to non-smoking partners). During the matching process, it may be also useful to know some general background or commonsense knowledge about the application domain, but still allowing exceptions (e.g., generally people of the same interests are happier in their marriages). With such a variety of knowledge (e.g., facts, conditionals, constraints, preferences, defaults, assumptions, exceptions), the desired nice properties of a matching may change as well.

With these motivations, we have studied one of the well-known matching problems, the Stable Marriage Problem (SMP), and its hard variants [12, 16]. In SMP, for a set of n men and n women, we are given the preferences of individuals: for each man, a complete ranking of the women is specified as preferred partners; similarly, for each woman, a complete ranking of the men is specified as preferred partners. The goal is to marry all men and women (i.e., to find n couples) in such a way that marriages are stable: no man and woman in different couples prefer each other to their partners.

In the hard variants of SMP, rankings may be incomplete (i.e., some partners are not acceptable) or may include ties (i.e., some partners are preferred equally). Also, optimal stable matchings are aimed with respect to different measures of fairness: sex-equality (preferences of men and women are considered to be equally important), egalitarian (preferences of every individual are considered to be equally important), minimum regret (minimizes the lowest ranking in marriages), maximum cardinality (minimizes the number of singles).

Since we focus on augmenting further types of knowledge to SMP with ties and incomplete preferences (SMPTI), due to its expressive formalism, we investigate the use of the knowledge representation and reasoning paradigm Answer Set Programming (ASP) [18, 20, 14, 23, 3, 4] to solve these problems. The idea of ASP is to represent a problem as a "program" whose models (called "answer sets" [11, 10]) correspond to the solutions. The answer sets for the given program can then be computed by special implemented systems called ASP solvers, such as Clingo [9]. Due to the continuous improvement of the ASP solvers and highly expressive representation language of ASP, ASP has been applied to a wide range of areas, including industrial applications [7].

We introduce elaboration-tolerant [19] formulations of SMPTI problems in a nondisjunctive ASP language, and use the ASP solver CLINGO to compute solutions. We also discuss formulations of different types of additional knowledge in ASP.

ASP has been investigated to solve SMPTI problems earlier [6] using disjunctive programs and the solver DLV [13]. We provide a comparison of the ASP-based approaches empirically.

2 Answer Set Programming

Answer Set Programming (ASP) [18, 20, 14, 23, 3, 4] is a knowledge representation and reasoning paradigm that is oriented towards combinatorial search problems as well as knowledge-intensive applications. The idea of ASP is to represent a problem and relevant knowledge as a "program" and to reason about the program by computing its models (called "answer sets" [11, 10]). These models characterize solutions of the problem, and can be computed by "ASP solvers", such as Clingo [9].

We consider ASP programs that consist of nondisjunctive rules of the form

$$\mathit{Head} \leftarrow A_1, \dots, A_m, \mathit{not}\ A_{m+1}, \dots, \mathit{not}\ A_n \tag{1}$$

where $n \geq m \geq 0$, *Head* is an atom or $\bot$, and each A_i is an atom. *Head* is called the *head* of the rules, and the formula $A_1, \dots, A_m, \mathit{not}\ A_{m+1}, \dots, \mathit{not}\ A_n$ is called the *body* of the rule. A rule is called a *fact* if $m = n = 0$ and a *(hard) constraint* if *Head* is $\bot$.

There are several useful utilities of ASP that are worth explaining here as they are used in our ASP formulations of SMPTI problems.

Cardinality expressions. When we represent a problem in ASP, we can use special constructs of the form $l\{A_1, \dots, A_k\}u$ (called *cardinality expressions*) where each

A_i is an atom and l and u are nonnegative integers denoting the lower and upper bounds [23]. Programs using these constructs can be viewed as abbreviations for programs that consist of rules of the form (1). Such an expression describes the subsets of the set $\{A_1, \ldots, A_k\}$ whose cardinalities are at least l and at most u.

Cardinality expressions can be used in heads of rules; then they generate many answer sets whose cardinality is at least l and at most u. For instance, the answer sets for the program

$$2\{p_1, ..., p_7\}4 \leftarrow \tag{2}$$

are subsets of the set $\{p_1, ..., p_7\}$ whose cardinality is at least 2 and at most 4.

Cardinality expressions can be also used in constraints; then they eliminate some of the answer sets. For instance, adding the constraint

$$\leftarrow 4\{p_1, ..., p_7\}$$

to (2) eliminates all of its answer sets that include subsets of $\{p_1, ..., p_7\}$ whose cardinality is at least 4.

Schematic variables. A group of rules that follow a pattern can be often described in a compact way using "schematic variables". For instance, we can write the program

$$p_i \leftarrow \textit{not}\ p_{i+1} \qquad (1 \leq i \leq 7)$$

as follows

$$\begin{array}{l} \textit{index}(1).\ \textit{index}(2).\ ...\ \textit{index}(7). \\ p(i) \leftarrow \textit{not}\ p(i+1), \textit{index}(i). \end{array}$$

The auxiliary predicate $\textit{index}(i)$ is introduced to describe the ranges of variables. ASP solvers compute an answer set for a given program that contains variables, after "grounding" the program. Variables can be also used "locally" to describe the list of formulas. For instance, the rule

$$1\{p_1, \ldots, p_7\}1$$

can be represented as follows:

$$1\{p(i) : \textit{index}(i)\}1.$$

Aggregates. ASP solvers support various aggregates, such as $\#\textit{count}$ (the number of elements in a set), $\#\textit{sum}$ (the sum of assigned weights of elements in a set), and $\#\textit{max}$ (the maximum weight among the assigned weights of elements in a set). For instance, suppose that, each atom $p(i)$ has a weight j; the weights can be defined by a set of facts of the form $\textit{weight}(i, j)$. Then the following constraint eliminates the answer sets where the total weight of all atoms of the form $p(i)$ included in the answer set is more than 5:

$$\leftarrow \#\textit{sum}\{j : p(i), \textit{index}(i), \textit{weight}(i, j)\} > 5.$$

Weak constraints. The ASP programs can be augmented with "weak constraints" [5]—expressions of the following form:

$$\overset{\frown}{\leftarrow} \textit{Body}(t_1, ..., t_n)[w@p, t_1, ..., t_n].$$

Here, $\mathit{Body}(t_1, ..., t_n)$ is a formula (as in the body of a rule (1)) with the terms $t_1, ..., t_n$. Intuitively, whenever an answer set for an ASP program satisfies $\mathit{Body}(t_1, ..., t_n)$, the tuple $\langle t_1, ..., t_n \rangle$ contributes a cost of w to the total cost function of priority p. The ASP solver tries to find an answer set with the minimum total cost. For instance, the following weak constraint

$$\overset{\backsim}{\leftarrow} p(i), p(i+1), index(i), index(i+1)[1@2, i] \tag{3}$$

instructs CLINGO to compute an answer set that does not include both $p(i)$ and $p(i+1)$, if possible. However, if CLINGO cannot find such an answer set, it is allowed to compute an answer set with these atoms $p(i)$ and $p(i+1)$ but with an additional cost of 1 per each such i. Weak constraints are considered by CLINGO according to their priorities.

3 Solving Stable Marriage Problems using ASP

We consider Stable Marriage Problems where rankings of individuals may contain Ties and may be Incomplete (SMPTI). It is defined by the following input:

- a set M of men,
- a set W of women,
- for every women $w \in W$, a partial function wrank_w that maps a man $m \in M$ to a positive integer, and
- for every men $m \in M$, a partial function mrank_m that maps a woman $w \in W$ to a positive integer.

Intuitively, for instance, $\mathit{wrank}_w(m) = i$ means that m is w's i'th most preferred partner. We say that a man m is acceptable for a woman w if $\mathit{wrank}_w(m)$ is defined. Similarly, a woman w is acceptable for a man m if $\mathit{mrank}_m(w)$ is defined.

An SMPTI problem $\langle M, W, \mathit{wrank}_w, \mathit{mrank}_m \rangle$ asks for a set S of marriages (w, m) ($w \in W$ and $m \in M$) such that

(S1) each woman (resp. man) is married to at most one man (resp. woman), and
(S2) no blocking pair of woman and man exists with respect to S.

An woman w is single with respect to S if there is no $x \in M$ such that $(w, x) \in S$; similarly, a man m is single with respect to S if there is no $x \in W$ such that $(x, m) \in S$.

A woman $w \in W$ and a man $m \in M$ is a blocking pair with respect to a set S of marriages if the following conditions hold:

(B1) w and m are acceptable to each other,
(B2) w and m are not married to each other (i.e., $(w, m) \notin S$),
(B3) w is single or w prefers m to her partner m' (i.e., $(w, m') \in S$, $\mathit{wrank}_w(m) < \mathit{wrank}_w(m')$),
(B4) m is single or m prefers w to his partner w' (i.e., $(w', m) \in S$): $\mathit{mrank}_m(w) < \mathit{mrank}_m(w')$.

Note that we consider weakly stable matchings. Also, it is assumed that marriage to an acceptable partner is preferred over being single.

Example 1. Consider, for instance, a variant of Example 4 of [6]. There are 3 women and 2 men, where the rankings are as follows:

$$\begin{array}{lll}
\mathit{wrank}_1(1) = 1 & \mathit{wrank}_1(2) = 1 & \\
\mathit{wrank}_2(1) = 1 & & \\
\mathit{wrank}_3(1) = 2 & \mathit{wrank}_3(2) = 1 & \\
 & & \\
\mathit{mrank}_1(1) = 1 & \mathit{mrank}_1(2) = 2 & \mathit{mrank}_1(3) = 2 \\
\mathit{mrank}_2(1) = 2 & \mathit{mrank}_2(2) = 1 &
\end{array}$$

Here there are three different stable sets of marriages:

$$\begin{array}{l}
S_1 = \{(3,1),(1,2)\} \\
S_2 = \{(2,1),(1,2)\} \\
S_3 = \{(1,1)\}
\end{array}$$

Note that Woman 3 is single in S_2, whereas only Man 1 and Woman 1 are married in S_3.

3.1 Solving SMPTI using ASP

We describe an SMPTI problem instance $\langle M, W, \mathit{wrank}_w, \mathit{mrank}_m\rangle$ by a set of facts using atoms of the forms $\mathit{woman}(w)$, $\mathit{man}(m)$, $\mathit{mrank}(m,w,i)$, and $\mathit{wrank}(w,m,i)$. For instance, the following facts describes wrank_3:

$$\begin{array}{l}
\mathit{wrank}(3,2,1). \\
\mathit{wrank}(3,1,2).
\end{array}$$

We start our ASP formulation of SMPTI by generating a set of pairs of woman and man such that every man m is matched to at most one woman that is acceptable for him:

$$\{\mathit{marry}(w,m) : \mathit{woman}(m), \mathit{acceptable}(w,m)\}1 \leftarrow \mathit{man}(m). \tag{4}$$

Here, acceptability is defined as follows:

$$\mathit{acceptable}(w,m) \leftarrow \mathit{wrank}(w,m,i), \mathit{mrank}(m,w,j).$$

Note that the condition (S1) for men is taken into account while generating pairs of woman and man by rules (4). We ensure the condition (S1) for women by adding the following constraint:

$$\leftarrow 2\{\mathit{marry}(w,m) : \mathit{man}(m)\}, \mathit{woman}(w).$$

With the rules above, CLINGO can generate a set S of marriages that satisfy (S1). For stability condition (S2), we need further constraints, i.e., (B1)–(B4). We consider four cases, and prevent blocking pairs accordingly.

Case 1 (w_2, m_1) is blocking S where $(w_1, m_1), (w_2, m_2) \in S$. To prevent such blocking pairs, we add the following constraints:

$$\begin{aligned}\leftarrow\ & \textit{man}(m_1), \textit{man}(m_2), \textit{woman}(w_1), \textit{woman}(w_2),\\ & \textit{marry}(w_1, m_1), \textit{marry}(w_2, m_2),\\ & \textit{mprefer}(m_1, w_2, w_1), \textit{wprefer}(w_2, m_1, m_2)\end{aligned}$$

where $\textit{mprefer}(m_1, w_2, w_1)$ describes that m_1 prefers w_2 to w_1:

$$\textit{mprefer}(m_1, w_2, w_1) \leftarrow \textit{mrank}(m_1, w_2, x), \textit{mrank}(m_1, w_1, x'), x < x'.$$

Here, $\textit{wprefer}(w_2, m_1, m_2)$ is defined similarly.

Case 2 (w_2, m) is blocking S where $(w_1, m) \in S$ and w_2 is single. To prevent such blocking pairs, we add the following constraints:

$$\begin{aligned}\leftarrow\ & \textit{man}(m), \textit{woman}(w_1), \textit{woman}(w_2),\\ & \textit{marry}(w_1, m), \textit{wsingle}(w_2),\\ & \textit{acceptable}(w_2, m), \textit{mprefer}(m, w_2, w_1)\end{aligned}$$

where $\textit{wsingle}(w_2)$ describes that woman w_2 is single:

$$\textit{wsingle}(w_2) \leftarrow \textit{woman}(w_2), \{\textit{marry}(w_2, m) : \textit{man}(m)\}0.$$

Case 3 (w, m_2) is blocking S where $(w, m_1) \in S$ and m_2 is single. This case is similar to Case 2.

Case 4 (w, m) is blocking S where w and m are single. To prevent such blocking pairs, we add the following constraints:

$$\leftarrow \textit{man}(m), \textit{woman}(w), \textit{acceptable}(w, m), \textit{msingle}(m), \textit{wsingle}(w).$$

With the ASP program described above, we can solve SMPTI problems. For instance, for Example 1, CLINGO computes all solutions as follows:

```
clingo version 4.5.4
Solving...
Answer: 1
marry(1,1) msingle(2) wsingle(2) wsingle(3)
Answer: 2
marry(1,2) marry(3,1) wsingle(2)
Answer: 3
marry(2,1) marry(1,2) wsingle(3)
SATISFIABLE

Models       : 3
Calls        : 1
Time         : 0.005s
CPU Time     : 0.000s
```

3.2 Finding Optimal Solutions to SMPTI

Among many stable marriages, some may be preferred better with respect to different measures of fairness. We consider four measures of fairness: sex-equality (preferences of men and women are considered to be equally important), egalitarian (preferences of every individual are considered to be equally important), minimum regret (minimizes the lowest ranking in marriages), maximum cardinality (minimizes the number of singles).

Let S be a solution to an SMPTI problem $\langle M, W, \mathit{wrank}_w, \mathit{mrank}_m\rangle$. We define the cost $c_i(S)$ of S for an individual $i \in M \cup W$ as follows: $c_i(S) = k$ if i is married to his/her k'th preferred partner in S.

Sex-equal SMPTI maximizes the sex equality of a stable set S of marriages by minimizing the following cost function $c(S)$:

$$c(S) = |\sum_{i \in M} c_i(S) - \sum_{i \in W} c_i(S)|.$$

To solve sex-equal SMPTI, we add the following weak constraints to the ASP program that represents SMPTI:

$$\overset{\smile}{\leftarrow} t = \#\mathit{sum}\{r_1 - r_2, w, m : \mathit{marry}(w,m), \mathit{mrank}(m,w,r_1), \mathit{wrank}(w,m,r_2)\}.[|t|]$$

Egalitarian SMPTI maximizes the total satisfaction of preferences of all individuals by maximizing the following cost function $c(S)$:

$$c(S) = \sum_{i \in M} c_i(S) + \sum_{i \in W} c_i(S).$$

To solve egalitarian SMPTI, we add the following weak constraints to the ASP program that represents SMPTI:

$$\overset{\smile}{\leftarrow} t = \#\mathit{sum}\{r_1 + r_2, w, m : \mathit{marry}(w,m), \mathit{mrank}(m,w,r_1), \mathit{wrank}(w,m,r_2)\}.[t]$$

Minimum regret SMPTI minimizes the maximum regret of S, defined by the following cost function $c(S)$:

$$c(S) = \mathit{max}\{c_i(S)\}_{i \in M \cup W}.$$

To solve minimum regret SMPTI, we add the following weak constraints to the ASP program that represents SMPTI:

$$\begin{aligned}\overset{\smile}{\leftarrow} t = \#\mathit{max}\{&r_1 : \mathit{marry}(w,m), \mathit{mrank}(m,w,r_1);\\ &r_2 : \mathit{marry}(w,m), \mathit{wrank}(w,m,r_2)\}.[t]\end{aligned}$$

The expression $\{r_1 : \mathit{marry}(w,m), \mathit{mrank}(m,w,r_1); r_2 : \mathit{marry}(w,m), \mathit{wrank}(w,m,r_2)\}$ can be understood as the union of $\{r_1 : \mathit{marry}(w,m), \mathit{mrank}(m,w,r_1)\}$ and $\{r_2 : \mathit{marry}(w,m), \mathit{wrank}(w,m,r_2)\}$.

Max cardinality SMPTI maximizes the number of marriages in S, by minimizing the number of singles:

$$c(S) = \mathit{count}\{m_i(S)\}_{i \in M \cup W}$$

where $m_i(S) = 1$ if i is not married in S; otherwise, $m_i(S) = 0$. To solve max cardinality SMPTI, we add the following weak constraints to the ASP program that represents SMPTI:

$$\overset{\sim}{\leftarrow} t = \#\mathit{count}\{m : \mathit{msingle}(m); w : \mathit{wsingle}(w)\}.[t]$$

Note that combinations of these optimizations are possible, by giving priorities to weak constraints: instead of specifying just the weight t of the weak constraints, we need to specify also their priorities p by an expression $w@p$.

4 Augmenting Domain-Specific Knowledge to SMPTI

While SMPTI problems are defined mainly over preferences of individuals and domain-independent optimality measures, real world applications may require specific knowledge about some individuals, general knowledge about their communities, traditions and social norms, and/or commonsense knowledge.

For instance, some women may prefer to be matched to nonsmoking partners. We can formulate these preferences utilizing weak constraints as follows:

$$\overset{\sim}{\leftarrow} \mathit{marry}(w, m), \mathit{wprefer\text{-}nosmoker}(w), \mathit{msmoker}(m).[1@4, w]$$

In Example 1, suppose that Woman 1 prefers to get married to a nonsmoker; she does not know who is not a nonsmoker. Suppose also that Man 1 is a smoker. Then, with the weak constraints above, CLINGO computes the solution S_3.

During the matching process, it may be also useful to know some general assumptions or commonsense knowledge about the application domain, but still allowing exceptions. For instance, suppose that, in a community, generally people of the same political views are happier in their marriages. This assumption can be expressed in ASP by a default as follows:

$$\begin{aligned}&\mathit{happier}(w, m) \leftarrow \mathit{not}\ \neg\mathit{happier}(w, m), \mathit{acceptable}(w, m),\\ &\qquad \mathit{wpolitics}(w, x), \mathit{mpolitics}(m, x)\end{aligned}$$

and then used as a criterion to maximize the number of happier married couples:

$$\overset{\sim}{\leftarrow} t = \#\mathit{count}\{w, m : \mathit{marry}(w, m), \mathit{happier}(w, m)\}.[N - t]$$

where N is the maximum number of marriages. In Example 1, suppose that Women 1 and 3 and Man 1 and 2 have the same political view X. Then, with the weak constraints above, CLINGO computes the solution S_1.

If there are exceptions, e.g., Man 1 does not get along with people with a political view X, they can be specified in ASP as well, e.g., as follows:

$$\neg\mathit{happier}(w, 1) \leftarrow \mathit{wpolitics}(w, X).$$

Due to nonmonotonocity of ASP, no rules above need to be changed. With this exception, CLINGO now computes the solution S_2.

Note that these additional knowledge an be combined with different measures of optimality above.

5 Experimental Evaluations

We have evaluated our ASP formulation empirically on some benchmarks, randomly generated for 20 women and 20 men. In these instances, the preferences of individuals may be incomplete: the percentages of specified rankings are %25, %50, %100. The preferences of individuals may have ties, with %0, %10, %20. Experiments are performed on a Linux server with 16 2.4GHz Intel E5-2665 CPU cores and 64GB memory. The ASP solver CLINGO 4.5.4 has been used in the experiments.

The results of these experiments are shown in Table 1. According to these results, without any optimizations, all instances of SMPTI can be solved in less than a second.

Minimum regret SMPTI and maximum cardinality SMPTI take similar amount of time, due to the similarities of the corresponding weak constraints. They consume slightly some more time compared to basic SMPTI.

Sex-equal SMPTI and Egalitarian SMPTI also take similar amount of time, due to the similarities of their representations with each other. On the other hand, they take much longer computation times as the percentage of completeness and the percentage of ties increase. This may be due to large sizes of summations used in weak constraints.

For a comparison, we have also experimented with the ASP formulation of SMPTI by De Clercq et al. [6]. The results are summarized in Table 2. As the percentage of completeness and the percentage of ties increase, we observe larger differences between the computation times with the two ASP formulations; for many instances, our ASP formulations takes less time.

Our ongoing studies involve experiments with larger instances. We plan to make the benchmark instances available online.

6 Conclusion

We have studied SMPTI and some of its optimization variants using ASP. We have observed benefits of declarative problem solving using ASP, while modeling these problems: cardinality expressions, aggregates, hard/weak constraints allow us to formalize the problems in a straightforward way. Moreover, the ASP representations of these problems are elaboration tolerant: to represent a variation of SMPTI, only new rules/constraints are added; no existing rule for SMPTI is modified while representing the optimization variants.

We have also investigated embedding various types of more domain-specific knowledge into formal representations of SMPTI problems using ASP: weak constraints allow us to formalize specific preferences of individuals, while defaults allow us to formalize general assumptions related to the subgroups of individuals. The elaboration tolerance of the ASP representations of SMPTI problems are preserved, as such different types of knowledge is added.

We have generated a set of benchmarks and evaluated our ASP formulation accordingly. For some SMPTI problems, ASP provides promising results. Our ongoing studies involve experimental evaluations with larger instances, and with alternative ASP formulations. We plan to make our benchmarks available, to be used for studies on SMPTI.

Table 1. Experiments with our ASP formulation: CPU times in seconds.

Input Size		Completeness (%)		Ties (%)		SexEqual	Egalitarian	MinRegret	MaxCardinality	SMPTI
Men	Women	Men	Women	Men	Women	(sec.)	(sec.)	(sec.)	(sec.)	(sec.)
20	20	25	25	0	0	0.010	0.020	0.000	0.000	0.000
20	20	25	25	0	10	0.020	0.040	0.000	0.010	0.010
20	20	25	25	0	20	0.010	0.020	0.000	0.000	0.010
20	20	25	25	10	10	0.010	0.020	0.000	0.010	0.010
20	20	25	25	10	20	0.010	0.040	0.010	0.000	0.000
20	20	25	25	20	20	0.020	0.030	0.000	0.010	0.010
20	20	25	50	0	0	0.030	0.110	0.010	0.010	0.000
20	20	25	50	0	10	0.050	0.160	0.010	0.020	0.010
20	20	25	50	0	20	0.020	0.110	0.010	0.010	0.000
20	20	25	50	10	10	0.040	0.080	0.000	0.010	0.010
20	20	25	50	10	20	0.020	0.100	0.010	0.010	0.010
20	20	25	50	20	20	0.050	0.200	0.010	0.010	0.020
20	20	25	100	0	0	0.120	0.580	0.020	0.020	0.020
20	20	25	100	0	10	0.390	0.670	0.020	0.020	0.020
20	20	25	100	0	20	0.280	0.860	0.020	0.010	0.010
20	20	25	100	10	10	0.110	0.430	0.020	0.020	0.020
20	20	25	100	10	20	0.130	0.570	0.020	0.010	0.020
20	20	25	100	20	20	0.140	0.860	0.020	0.020	0.010
20	20	50	50	0	0	0.120	0.480	0.010	0.020	0.010
20	20	50	50	0	10	0.250	1.390	0.020	0.020	0.020
20	20	50	50	0	20	0.140	0.550	0.010	0.020	0.010
20	20	50	50	10	10	0.100	0.410	0.020	0.020	0.010
20	20	50	50	10	20	0.300	1.200	0.020	0.010	0.020
20	20	50	50	20	20	0.680	1.350	0.020	0.020	0.010
20	20	50	100	0	0	5.400	6.810	0.030	0.040	0.030
20	20	50	100	0	10	1.490	8.410	0.040	0.040	0.040
20	20	50	100	0	20	0.830	8.530	0.030	0.020	0.040
20	20	50	100	10	10	2.830	4.190	0.020	0.030	0.040
20	20	50	100	10	20	1.560	12.020	0.030	0.030	0.030
20	20	50	100	20	20	10.690	13.990	0.040	0.040	0.040
20	20	100	100	0	0	1.810	55.060	0.110	0.100	0.110
20	20	100	100	0	10	24.950	40.200	0.120	0.100	0.110
20	20	100	100	0	20	86.500	54.760	0.130	0.110	0.110
20	20	100	100	10	10	23.340	90.570	0.110	0.110	0.110
20	20	100	100	10	20	74.040	49.750	0.110	0.110	0.100
20	20	100	100	20	20	39.990	76.780	0.110	0.130	0.100

Table 2. Experiments with the ASP formulation of [6]: CPU times in seconds.

Input Size		Completeness (%)		Ties (%)		SexEqual	Egalitarian	MinRegret	MaxCardinality	SMPTI
Men	Women	Men	Women	Men	Women	(sec.)	(sec.)	(sec.)	(sec.)	(sec.)
20	20	25	25	0	0	0.520	0.030	0.000	0.000	0.000
20	20	25	25	0	10	1.600	0.070	0.000	0.000	0.000
20	20	25	25	0	20	0.920	0.050	0.000	0.000	0.000
20	20	25	25	10	10	0.640	0.040	0.000	0.000	0.000
20	20	25	25	10	20	1.200	0.060	0.000	0.000	0.000
20	20	25	25	20	20	1.170	0.050	0.000	0.000	0.000
20	20	25	50	0	0	3.160	0.110	0.010	0.010	0.000
20	20	25	50	0	10	4.090	0.160	0.010	0.010	0.000
20	20	25	50	0	20	3.180	0.120	0.020	0.000	0.000
20	20	25	50	10	10	2.380	0.100	0.020	0.000	0.000
20	20	25	50	10	20	2.740	0.110	0.010	0.000	0.000
20	20	25	50	20	20	4.460	0.160	0.010	0.000	0.000
20	20	25	100	0	0	13.530	0.470	0.040	0.020	0.000
20	20	25	100	0	10	14.730	0.520	0.040	0.020	0.000
20	20	25	100	0	20	17.550	0.530	0.060	0.020	0.000
20	20	25	100	10	10	9.360	0.350	0.040	0.010	0.000
20	20	25	100	10	20	16.650	0.640	0.040	0.020	0.000
20	20	25	100	20	20	18.370	0.640	0.050	0.020	0.000
20	20	50	50	0	0	15.170	0.480	0.040	0.000	0.000
20	20	50	50	0	10	18.830	0.600	0.050	0.020	0.000
20	20	50	50	0	20	9.820	0.330	0.040	0.010	0.000
20	20	50	50	10	10	9.370	0.310	0.030	0.010	0.000
20	20	50	50	10	20	16.750	0.530	0.040	0.020	0.000
20	20	50	50	20	20	24.970	0.930	0.060	0.020	0.000
20	20	50	100	0	0	58.620	2.030	0.150	0.040	0.000
20	20	50	100	0	10	71.770	2.610	0.190	0.050	0.000
20	20	50	100	0	20	65.890	2.200	0.170	0.050	0.000
20	20	50	100	10	10	67.940	2.750	0.180	0.040	0.000
20	20	50	100	10	20	82.680	4.200	0.190	0.050	0.010
20	20	50	100	20	20	93.730	6.600	0.250	0.050	0.000
20	20	100	100	0	0	369.640	24.670	1.110	0.220	0.000
20	20	100	100	0	10	386.900	21.800	1.090	0.230	0.010
20	20	100	100	0	20	341.700	28.170	1.010	0.220	0.010
20	20	100	100	10	10	414.580	25.710	1.210	0.220	0.000
20	20	100	100	10	20	324.570	26.660	1.110	0.240	0.000
20	20	100	100	20	20	304.990	17.920	1.040	0.220	0.000

References

1. Alkan, A., Gale, D.: Stable schedule matching under revealed preference. Journal of Economic Theory **112**(2), 289–306 (2003)
2. Alkan, A., Moulin, H.: Mathematical theories of allocation of discrete resources: equilibria, matchings, mechanisms. Elsevier (2003)
3. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011)
4. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming: An introduction to the special issue. AI Magazine **37**(3), 5–6 (2016)
5. Buccafurri, F., Leone, N., Rullo, P.: Enhancing disjunctive datalog by constraints. IEEE Trans. Knowl. Data Eng. **12**(5), 845–860 (2000)
6. De Clercq, S., Schockaert, S., De Cock, M., Nowe, A.: Solving stable matching problems using answer set programming. Theory and Practice of Logic Programming **16**(3), 247–268 (2016)
7. Erdem, E., Gelfond, M., Leone, N.: Applications of answer set programming. AI Magazine **37**(3), 53–68 (2016)
8. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. American Mathematical Monthly **69**, 9–15 (1962)
9. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Clingo = ASP + control: Preliminary report. In: Proc. of ICLP (Technical Communications) (2014)
10. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of International Logic Programming Conference and Symposium, pp. 1070–1080 (1988)
11. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9**, 365–385 (1991)
12. Kato, A.: Complexity of the sex-equal stable marriage problem. Japan Journal of Industrial and Applied Mathematics (JJIAM) **10**, 1–19 (1993)
13. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Trans. Comput. Logic **7**, 499–562 (2006)
14. Lifschitz, V.: Answer set programming and plan generation. Artif. Intell. **138**, 39–54 (2002)
15. Manlove, D.: Stable marriage with ties and unacceptable partners. Technical report, University of Glasgow, Department of Computing Science (1999)
16. Manlove, D., Irving, R.W., Iwama, K., Miyazaki, S., Morita, Y.: Hard variants of stable marriage. Theoretical Computer Science **276**(1–2), 261–279 (2002)
17. Manlove, D.F., OMalley, G.: Paired and altruistic kidney donation in the UK: algorithms and experimentation. ACM Journal of Experimental Algorithmics **19** (2014)
18. Marek, V., Truszczynski, M.: Stable models and an alternative logic programming paradigm. In: The Logic Programming Paradigm: a 25-Year Perspective, pp. 375–398. Springer Verlag (1999)
19. McCarthy, J.: Elaboration tolerance. In: Proc. of CommonSense (1998)
20. Niemelae, I.: Logic programs with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence **25**, 241–273 (1999)
21. Roth, A.E., Sonmez, T., Unver, M.U.: Pairwise kidney exchange. Journal of Economic Theory **125**, 151–188 (2005)
22. Roth, A.E., Sotomayor, M.A.O.: Two-sided matching: a study in game-theoretic modeling and analysis. Econometric Society Monographs **18** (1990)
23. Simons, P., Niemelae, I., Soininen, T.: Extending and implementing the stable model semantics. Artif. Intell. **138**(1), 181–234 (2002)