TPLP **20** (6): 911–925, 2020.

doi:10.1017/S1471068420000277

A General Framework for Stable Roommates Problems using Answer Set Programming∗

ESRA ERDEM and MÜGE FIDAN

Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey
{*esraerdem,mugefidan*}*@sabanciuniv.edu*

DAVID MANLOVE and PATRICK PROSSER

School of Computing Science, University of Glasgow, Glasgow, UK
{*david.manlove,patrick.prosser*}*@glasgow.ac.uk*

submitted 7 August 2020; accepted 10 August 2020

Abstract

The Stable Roommates problem (SR) is characterized by the preferences of agents over other agents as roommates: each agent ranks all others in strict order of preference. A solution to SR is then a partition of the agents into pairs so that each pair shares a room, and there is no pair of agents that would block this matching (i.e., who prefers the other to their roommate in the matching). There are interesting variations of SR that are motivated by applications (e.g., the preference lists may be incomplete (SRI) and involve ties (SRTI)), and that try to find a more fair solution (e.g., Egalitarian SR). Unlike the Stable Marriage problem, every SR instance is not guaranteed to have a solution. For that reason, there are also variations of SR that try to find a good-enough solution (e.g., Almost SR). Most of these variations are NP-hard. We introduce a formal framework, called SRTI-ASP, utilizing the logic programming paradigm Answer Set Programming, that is provable and general enough to solve many of such variations of SR. Our empirical analysis shows that SRTI-ASP is also promising for applications.

KEYWORDS: stable roommates problem, answer set programming, declarative problem solving

1 Introduction

The Stable Roommates problem (Gale and Shapley 1962) (SR) is a matching problem (well-studied in Economics and Game Theory) characterized by the preferences of an even number n of agents over other agents as roommates: each agent ranks all others in strict order of preference. A solution to SR is then a partition of the agents into pairs that are *acceptable* to each other (i.e., they are in the preference lists of each other), and the matching is *stable* (i.e., there exist no two agents who prefer each other to their roommates, and thus *block* the matching).

SR is an interesting computational problem, not only due to its applications (e.g., for pairing in large-scale chess competitions (Kujansuu et al. 1999), for campus house allocation (Arkin et al. 2009), pairwise kidney exchange (Roth et al. 2005), creating partnerships in P2P networks (Gai et al. 2007)) but also due to its computational properties described below.

∗ This work has been partially supported by Sabanci University IRP Grant, and the Scottish Informatics and Computer Science Alliance DVF Programme. The third and fourth authors were supported by grants EP/P028306/1 and EP/P026842/1 from the Engineering and Physical Sciences Research Council, respectively.

Incomplete preference lists with ties. Upon a question posed by Knuth (1997) in 1976 about the existence of an algorithm for SR, Irving (1985) developed a linear-time algorithm for SR. Meanwhile, researchers have started investigating variations of SR motivated by further observations and applications. For instance, in practice (like large-scale chess tournaments), agents may find it difficult to rank a large number of alternatives in strict order of preference. With such motivations, SR has been studied with incomplete preference lists (SRI) (Gusfield and Irving 1989), with preference lists including ties (SRT) (Ronn 1990), and with incomplete preference lists including ties (SRTI) (Irving and Manlove 2002). Interestingly, some of these slight variations (i.e., the existence of a stable matching in SRT and SRTI) are proven to be NP-complete (Table 1).

Stable and more fair solutions. With the motivation of finding more fair stable solutions, variations of SR have been studied. For instance, Egalitarian SR aims to maximize the total satisfaction of preferences of all agents; it is NP-hard (Feder 1992). Rank Maximal SRI aims to maximize the number of agents matched with their first preference, and then, subject to this condition, to maximize the number of agents matched with their second preference, and so on; it is also NP-hard (Cooper 2020).

Not stable but good-enough solutions. As first noted by Gale and Shapley (1962), unlike the Stable Marriage problem (SM), there is no guarantee to find a solution to every SR problem instance (i.e., there might be no stable matching). When an SR instance does not have a stable solution, variations of SR have been studied to find a good-enough solution. Almost SR aims to minimize the total number of blocking pairs (i.e., pairs of agents who prefer each other to their roommates); it is NP-hard (Abraham et al. 2005).

Alongside these interests in SR, some exact methods and software (SRItoolkit 2019; MatchingToolkit 2020) have been developed to solve SR and SRI (both solvable in poly-time) using Constraint Programming (CP) (Prosser 2014), and based on Irving's algorithm (Irving 1985). However, to the best of the authors' knowledge, there is no exact method (except for the enumeration based method for Egalitarian SRI) and implementation, that provides a solution to any intractable variation of SR, described in three groups above.

Our Contributions. We introduce a formal framework and its implementation, called SRTI-ASP, that are general enough to provide solutions to all variations of SR mentioned above, including the intractable decision/optimization versions: SRT, SRTI, Egalitarian SRTI, Rank Maximal SRTI, Almost SRTI. SRTI-ASP provides a flexible framework to study variations of SR.

SRTI-ASP utilizes a logic programming paradigm, called Answer Set Programming (ASP) (Brewka et al. 2016), to declaratively solve stable roommates problems. We represent SRI and its variations in the expressive formalism of ASP, and SRTI-ASP computes models of these formulations using the ASP solver CLINGO (Gebser et al. 2011). For each variation of SR, given a problem instance, SRTI-ASP returns a solution (or all solutions) if one exists; otherwise, it returns that the problem does not have a solution. We prove that SRTI-ASP is sound and complete (Theorem 1).

We have evaluated SRTI-ASP over different sizes of SRI instances (randomly generated with the software (SRItoolkit 2019), called SRI-CP from now on) to understand its scalability, as the input size, and the degree of completeness of preference lists increase. We have developed a method to add ties to these instances, and empirically analyzed the scalability of SRTI-ASP on SRTI instances as well.

We have compared SRTI-ASP with SRI-CP, over SRI instances. We have also investigated

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

the use of SRI-CP to solve Egalitarian SRI, Rank Maximal SRI and Almost SRI based on enumeration-based brute-force methods, and compared SRTI-ASP with these methods.

In addition, we have compared SRTI-ASP with the ASP method proposed by Amendola (2018) for SR (called SR-AF from now on) based on Argumentation Framework (AF) (Dung 1995), over SR instances.

Table 1. *Summary of the complexities of SR problems*

Problem	Complexity
SR	P (Irving 1985)
SRI	P (Gusfield and Irving 1989)
SRTI (super)	P (Irving and Manlove 2002)
SRTI (strong)	P (Kunysz 2016; Scott 2005)
SRT (weak) (and thus SRTI (weak))	NP-complete (Ronn 1990, Thm 1.1, Prop 2.2)
SRTI (weak)	NP-complete* (Irving et al. 2009, Thm 5)
Egalitarian SR	NP-hard (Feder 1992, Thm 8.3)
Egalitarian SRI	NP-hard* (Cseh et al. 2019, Cor 4)
Almost SR (and thus SRT (weak))	NP-hard (Abraham et al. 2005, Thm 1)
Almost SRI	NP-hard* (Biró et al. 2012, Thm 1)

* for short lists of size ≤ 3

2 Stable Roommates Problems

Let us start with defining the Stable Roommate problem with Incomplete lists (SRI). Let A be a finite set of agents. For every agent $x \in A$, let $\prec_x$ be a strict and total ordering of preferences over a subset A_x of $A \backslash \{x\}$. We refer to $\prec_x$ as agent x's preference list. For two agents y and z, we denote by $y \prec_x z$ that x prefers y to z. Since the ordering of preferences is strict and total, for every agent $x \in A$, for every two distinct agents y and z in A_x, either $y \prec_x z$ or $z \prec_x y$. Note that the preferences of agents with respect to $\prec_x$ are transitive and asymmetric. If an agent x is in y's preference list, then x is called *acceptable* to y. We denote by $\prec$ the collection of all preference lists.

A *matching* for a given SRI instance is a function $M : A \mapsto A$ such that, for all $\{x, y\} \subseteq A$ such that $x \in A_y$ and $y \in A_x$, $M(x) = y$ if and only if $M(y) = x$. If agent x is mapped to itself, we then say he/she is *single*.

A matching M is *blocked* by a pair $\{x, y\} \subseteq A$ $(x \neq y)$ if

B1 both agents x and y are acceptable to each other,
B2 x is single with respect to M, or $y \prec_x M(x)$, and
B3 y is single with respect to M, or $x \prec_y M(y)$.

A matching for SRI is called *stable* if it is not blocked by any pair of agents. Fig. 1 illustrates three examples for SRI.

The Stable Roommates problem (SR) is a special case of SRI where the preference orderings are strict and complete (i.e., for every agent $x \in A$, $A_x = A \setminus \{x\}$), and $|A|$ is even.

Ties. The Stable Roommates problem with Ties and Incomplete Lists (SRTI) is a variation of SRI where the preference lists are partial orderings and where incomparability is transitive. In this

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

sri7	sri4	sri8
a: $b\ e\ d\ f\ g$ b: $c\ f\ a\ g\ e$ c: $d\ g\ b\ e\ f\ a$ d: $a\ c\ e\ f\ g$ e: $f\ a\ b\ c\ d$ f: $g\ b\ e\ c\ d\ a$ g: $c\ f\ d\ a\ b$	a: $b\ c\ d$ b: $c\ a\ d$ c: $a\ b\ d$ d: $a\ b\ c$	a: $c\ e\ f\ g\ d\ h$ b: $d\ f\ h\ c\ g$ c: $a\ b\ f\ h\ e\ d$ d: $h\ g\ e\ a\ b\ c$ e: $g\ c\ b\ d\ a\ f$ f: $e\ a\ g\ c\ h\ b$ g: $f\ h\ d\ b\ c$ h: $b\ d\ a\ e\ f$

Fig. 1. Three SRI instances. sri7 has a single stable solution $\{\{a,b\},\{c,d\},\{f,g\},\{e\}\}$. sri4 has no solutions, since each possible matching is blocked (e.g., $\{\{a,c\},\{b,d\}\}$ is blocked by $\{a,b\}$). sri8 has two stable matchings, M_1:$\{\{a,c\},\{b,h\},\{d,e\},\{f,g\}\}$ and M_2:$\{\{a,c\},\{b,h\},\{d,g\},\{e,f\}\}$.

context, ties correspond to indifference in the preference lists: an agent x is *indifferent* between the agents y and z, denoted by $y \sim_x z$, if $y \not\prec_x z$ and $z \not\prec_x y$. There are three levels of stability (Irving and Manlove 2002): *weak stability*, *strong stability*, and *super stability*. We will focus on *weak stability* in this paper, since it is a harder problem compared to the other two versions (Table 1). Relative to weak stability, a pair $\{x,y\}$ of agents *blocks* a matching M if conditions B1–B3 hold.

The Stable Roommates problem with Ties (SRT) is a special case of SRTI where the preference ordering of each agent x is over $A \setminus \{x\}$ and complete, and $|A|$ is even.

Note that, while the problems SR and SRI are in P, SRT and SRTI under weak stability are NP-complete (Table 1).

Fairness. When an SRI instance has many stable matchings, it may be useful to identify a stable matching that is fair to all agents. Different fairness criteria on top of stability have led to optimization variations of SRI.

Let $\mathscr{M}$ denote the set of all stable matchings of a given SRI instance $(A, \prec)$. For every agent x and every agent $y \in A_x$, let $rank(x,y)$ denote the rank of agent y in the preference list A_x of agent x. We assume that agents prefer matching with a roommate: for every agent x, let $rank(x,x)$ be a number larger than $rank(x,y)$ for every $y \in A_x$.

Egalitarian SRI aims to maximize the total satisfaction of preferences of all agents. Let M be a matching. For every agent x, we define the satisfaction $c_M(x)$ of x's preferences with respect to M as follows: $c_M(x)=I$ if $rank(x,M(x))=I$. Then the total satisfaction of preferences of all agents is defined as follows: $c(M) = \sum_{x \in A} c_M(x)$. Note that for SRI, all matching M have the same number of contributions of $rank(x,x)$ values to $c(M)$. Since the preferred agents have lower rankings, the total satisfaction of preferences of all agents is maximized when $c(M)$ is minimized. Then, a matching $M \in \mathscr{M}$ with the minimum $c(M)$ is *egalitarian.*

Rank Maximal SRI considers different fairness criterion: it aims to maximize the number of agents matched with their first preferences, and then, subject to this condition, to maximize the number of agents matched with their second preference, and so on. We start with the set $\mathscr{M}$ of all matchings of a given SRI instance $(A, \prec)$, and define a series of subsets $\mathscr{M}_{max}(i)$ of these matchings where the maximum number of agents are matched with their i'th preferences:

$$\mathscr{M}_{max}(0) = \mathscr{M}$$
$$\mathscr{M}_{max}(i) = \{M \in \mathscr{M}_{max}(i-1) : 1 \leq i \leq |A|-1,\ \forall\, M' \in \mathscr{M}_{max}(i-1) \text{ s.t. } M' \neq M$$
$$|\{x \in A : rank(x,M(x))=i, x \neq M(x)\}| \geq |\{x \in A : rank(x,M'(x))=i, x \neq M(x)\}|\}.$$

Then, a matching $M \in \mathscr{M}_{max}(|A|-1)$ is *rank-maximal*.

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

Consider the SRI instance sri8 illustrated in Fig. 1, with two stable matchings. Stable matching M_1 is egalitarian and M_2 is rank-maximal.

Almost stable. Unlike the Stable Marriage problem, there is no guarantee to find a solution to every SRI problem instance (cf. sri4 in Fig. 1). When an SRI instance does not have a stable matching, further variations of SRI have been studied to find a good-enough solution.

Almost SRI aims to minimize the total number of blocking pairs. Let $bp_M(x,y)$ denote the set of blocking pairs of a given matching M. A matching $M \in \mathscr{M}$ is *almost stable* if it is blocked by the minimum number $|bp_M(x,y)|$ of pairs.

3 Answer Set Programming

SRTI-ASP utilizes Answer Set Programming (ASP) (Brewka et al. 2016) to declaratively solve stable roommates problems. The idea of problem solving with ASP is (1) to represent the given problem by a program whose answer sets (Gelfond and Lifschitz 1988; Gelfond and Lifschitz 1991) characterize the solutions of the problem, and (2) to solve the problem using answer set solvers, like CLINGO (Gebser et al. 2011).

Why ASP? We use ASP as an underlying paradigm for modeling and solving stable roommates problems for the following reasons. (1) Deciding whether a program in ASP has an answer set is NP-complete (Dantsin et al. 2001), so ASP is expressive enough for solving hard SR problems. (2) ASP has expressive languages with a rich set of utilities, such as nondeterministic choices, hard constraints, weighted weak constraints with priorities, and thus allow us to easily formulate different variations of SR. (3) Efficient ASP solvers, like CLINGO, supports these utilities. (4) Such an elaboration tolerant (McCarthy 1998) representation framework and flexible software environment are useful in studying and understanding variations of SR in different applications. (5) Due to declarative problem solving in the formal framework of ASP, we can easily prove the soundness and completeness of SRTI-ASP (see Theorem 1).

Programs in ASP Let us briefly describe the syntax of programs and useful constructs used in the paper. We consider ASP programs that consist of rules of the form

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

where $n \geq m \geq 0$, *Head* is an atom or $\bot$, and each A_i is an atom. A rule is called a *fact* if $m = n = 0$ and a *(hard) constraint* if *Head* is $\bot$.

Cardinality expressions are special constructs of the form $l\{A_1, \ldots, A_k\}u$ where each A_i is an atom and l and u are nonnegative integers denoting the lower and upper bounds (Simons et al. 2002). 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.

Schematic variables A group of rules that follow a pattern can be often described in a compact way using "schematic variables". For instance, the cardinality expression $1\{p_1, \ldots, p_7\}1$ can be represented as $1\{p(i) : index(i)\}1$, along with a definition of $index(i)$ that describes the ranges of variables: $index(1..7)$.

Weighted weak constraints with priorities The ASP programs can be augmented with "weak constraints"—expressions of the following form (Buccafurri et al. 2000):

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

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

Here, $Body(t_1,...,t_n)$ is a formula (as in the body of a rule) with the terms $t_1,...,t_n$. Intuitively, whenever an answer set for a program satisfies $Body(t_1,...,t_n)$, 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

$$\widetilde{\leftarrow} p(i), p(i+1), index(i), index(i+1)[1@2, i]$$

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.

4 Solving SRI using ASP

We formalize the input $I = (A, \prec)$ of an SRI instance in ASP by a set F_I of facts using atoms of the forms $agent(x)$ ("x is an agent in A") and $prefer2(x,y,z)$ ("agent x prefers agent y to agent z, i.e., $y \prec_x z$ "). For instance, the preference list of agent a in sri4 of Fig. 1 is described by the following facts: $prefer2(a,b,c).\ prefer2(a,c,d).$

For every agent x, since x prefers being matched with a roommate y in A_x instead of being single, for every $y \in A_x$, we also add facts of the form $prefer2(x,y,x)$. For the example above, the input also includes the facts: $prefer2(a,b,a).\ prefer2(a,c,a).\ prefer2(a,d,a).$

In the ASP formulation P of SRI, the variables x, y, z and w denote agents in A. The program P starts with the definition of preferences of agents with respect to $\prec_x$:

$$\begin{array}{l} prefer(x,y,z) \leftarrow prefer2(x,y,z). \\ prefer(x,y,z) \leftarrow prefer2(x,y,w), prefer(x,w,z). \end{array} \tag{2}$$

The first rule expresses that being single is the least preferred option. The second rule expresses that the preference relation is transitive.

Based on the preferences of agents, we define the concept of acceptability for each agent:

$$\begin{array}{l} accept(x,y) \leftarrow prefer(x,y,\_). \\ accept(x,y) \leftarrow prefer(x,\_,y). \end{array} \tag{3}$$

and the concept of mutual acceptability:

$$accept2(x,y) \leftarrow accept(x,y), accept(y,x). \tag{4}$$

The output $M : A \mapsto A$ of an SRI instance is characterized by atoms of the form $room(x,y)$ ("agents x and y are roommates"). The ASP formulation P of SRI first generates pairs of roommates. For every agent x, exactly one mutual acceptable agent y is nondeterministically chosen as $M(x)$ by the choice rules:

$$1\{room(x,y){:}agent(y), accept2(x,y)\}1 \leftarrow agent(x). \tag{5}$$

Here, the roommate relation is symmetric:

$$\leftarrow room(x,y), not\ room(y,x). \tag{6}$$

The agents who are not matched with a roommate are single agents:

$$single(x) \leftarrow room(x,x). \tag{7}$$

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

Then, the stability of the generated matching is ensured by the hard constraints:

$$\leftarrow block(x,y) \qquad (x \neq y). \tag{8}$$

Here, atoms of the form $block(x,y)$ describe the blocking pairs (i.e., conditions B1–B3):

$$\begin{array}{l} block(x,y) \leftarrow accept2(x,y), single(x), single(y), not\ room(x,y). \quad (x \neq y) \\ block(x,y) \leftarrow accept2(x,y), single(x), like(y,x), not\ room(x,y). \quad (x \neq y) \\ block(x,y) \leftarrow accept2(x,y), like(x,y), single(y), not\ room(x,y). \quad (x \neq y) \\ block(x,y) \leftarrow accept2(x,y), like(x,y), like(y,x), not\ room(x,y). \quad (x \neq y) \end{array} \tag{9}$$

where $x \neq y$ in each rule, and atoms $like(x,y)$ describe that agent x prefers agent y to her/his roommate $x' = M(x)$:

$$like(x,y) \leftarrow room(x,x'), prefer(x,y,x'). \quad (x' \neq y) \tag{10}$$

Given the ASP formulation P whose rules are described above and the ASP description F_I of an SRI instance I, the ASP solver CLINGO generates a stable matching (or all stable matchings), if one exists; otherwise, it returns that there is no solution. This is possible since the ASP program P (i.e., (2)–(10)) is sound and complete.

Theorem 1
Given an SRI instance $I = (A, \prec)$, for each answer set S for $P \cup F_I$, the set of atoms of the form $room(x,y)$ in S encodes a stable matching $M : A \mapsto A$ to the SRI problem instance. Conversely, each stable matching for the given SRI instance corresponds to a single answer set for $P \cup F_I$.

Proof
First, we show that the answer set for (2)–(4) correctly describes the acceptability relation.

1. Due to Proposition 4 of Erdem and Lifschitz (2003) about the correctness of the transitive closure definition, the answer set X_0 for (2) correctly defines preferences of each agent x (by means of atoms of the $prefer(x,y,z)$).
2. Due to Proposition 3 of Erdogan and Lifschitz (2004), adding (3) to (2) conservatively extends X_0 to X_1, which also describes the preference lists for each agent x (by means of atoms of the $accept(x,y)$).
3. Due to Proposition 3 of Erdogan and Lifschitz (2004), adding (4) to $(2) \cup (3)$ conservatively extends X_1 to X.
4. The answer set X describes the acceptability of every pair of agents x and y to each other (by means of atoms of the $accept2(x,y)$).

Next, we show that the answer set for (2)–(6) correctly characterizes a matching.

1. We use the splitting set theorem (Erdogan and Lifschitz 2004; Lifschitz and Turner 1994): Let Π be the program (2)–(5), and the splitting set U be the set of atoms of the form $prefer(x,y,z)$, $accept(x,y)$, and $accept2(x,y)$.
2. The bottom $b_U(\Pi)$ consists of (2)–(4), and the top part consists of rules (5).
3. Every answer set Y for the top part $e_U(\Pi \setminus b_U(\Pi), X)$ evaluated with respect to X, describes a function, via atoms of the form $room(x,y)$, which maps every agent x to exactly one agent y so that x and y are acceptable to each other (when $accept2(x,y) \in X$). Moreover, every such mapping can be characterized by a unique answer set for $e_U(\Pi \setminus b_U(\Pi), X)$.
4. According to the splitting set theorem, $X \cup Y$ is an answer set for (2)–(5).

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

5. Then, $X \cup Y$ describes a mapping between acceptable pairs of agents. The symmetry of this mapping is guaranteed by (6), using Proposition 2 of Erdogan and Lifschitz (2004), leading to a matching.
6. Therefore, there is a one-to-one correspondence between the answer sets for (2)–(6) and the matchings between acceptable pairs of agents.

Next, using Proposition 3 of Erdogan and Lifschitz (2004) three times, we show that adding definitions (7), (10), and (9) to (2)–(6) one by one conservatively extends the answer sets for (2)–(6), by describing singles (by means of atoms of the $single(x)$), preferences y of every agent x over her/his roommate (by means of atoms of the $like(x,y,z)$), and then blocking pairs (by means of atoms of the $block(x,y)$).

Finally, we show that stability is guaranteed by (8), using Proposition 2 of Erdogan and Lifschitz (2004). Therefore, there is a one-to-one correspondence between every answer set S for (2)–(8) and every stable matching. □

Ties (weak stability). As noted by Cseh (2019), relative to weak stability, stability in SRTI instances can be defined in exactly the same way as for SRI. Therefore, we can use the SRI formulation P to solve SRTI instances too.

Fairness. Let us describe the ranks of agents by a set of facts using atoms of the form $rank(x,y,i)$ ("the rank of agent y according to agent x's preferences is i"). For each agent x, since x's preference ordering $\prec_x$ is total, we can define the ranks as follows: the i'th agent in the preference list A_x of x has rank i, and x has a rank larger than the ranks of i.

$$rank(x,b,1) \leftarrow \#count\{a : prefer(x,a,b), a \neq b\}{=}0, accept(x,b)$$
$$rank(x,b,i) \leftarrow rank(x,a,i{-}1), prefer(x,a,b),$$
$$\#count\{c : prefer(x,a,c), prefer(x,c,b), c \neq a, c \neq b\}{=}0 \quad (a \neq b, i > 1)$$

Egalitarian SRI aims to maximize the total satisfaction of preferences of all agents by a matching M. The satisfaction $c_M(x)$ of an agent x's preferences with respect to M is defined as the rank of $M(x)$. Since more preferred agents have lower ranks, the total satisfaction of preferences of all agents is maximized when $c(M){=}\sum_{x \in A} c_M(x)$ is minimized. Therefore, to solve Egalitarian SRI, we simply add to the SRI formulation P, the weighted weak constraints:

$$\tilde{\leftarrow} room(x,y), rank(x,y,r). \ [r@1,x]$$

which instruct CLINGO to minimize the sum of the ranks r of roommates.

Rank Maximal SRI tries to maximize the number of agents matched with their first preferences, and, subject to this condition, tries to maximize the number of agents matched with their second preferences, and so on. Such an iterative definition can be modeled elegantly by the following weak constraints:

$$\tilde{\leftarrow} room(x,y), rank(x,y,r). \ [-1@|A|-r,x,y] \qquad (x \neq y)$$

Note that the priorities of these weak constraints are defined as $|A|-r$ for every pair of roommates $\{x,y\}$ $(x \neq y)$, where $rank(x,y) = r$. As the rank changes from 1 to $|A|-1$, the priority decreases. The ASP solver CLINGO handles weak constraints with respect to their priorities. On the other hand, note that the weights of these weak constraints are specified as -1 for every pair of roommates $\{x,y\}$ $(x \neq y)$. Therefore, CLINGO first considers the highest priority $|A|-1$, and tries to minimize the total weights of agents matched with their first preferences. Then, CLINGO considers the next highest priority $|A|-2$, and further tries to minimize the total weights of agents

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

matched with their second preferences, and so on. In this way, CLINGO finds a rank maximal stable matching.

Almost stable. Almost SRI aims to minimize the total number of blocking pairs for a matching M. For that, we simply replace the hard constraint (8) that ensures stability, with the following weak constraints in our ASP formulation P of SRI:

$$\tilde{\leftarrow} block(x,y).\ [1@1,x,y] \quad (x \neq y)$$

Elaboration tolerance. According to McCarthy (1998), a representation is elaboration tolerant to the extent that it is convenient to modify a set of formulas expressed in the formalism to take into account new phenomena, and the simplest kind of elaboration is the addition of new formulas. In that sense, our representation P of SRI is elaboration tolerant to variations of SRI, since the program P is not changed at all (e.g., we add new rules to P for Egalitarian SRI) or it is changed minimally (e.g., we replace a hard constraint by a weak constraint for Almost SRI).

5 Experimental Evaluations

We have experimentally evaluated SRTI-ASP to understand its scalability over intractable SRTI problems, and how it compares with two closely related methods over tractable SR problems.

Setup. We have generated instances using the random instance generator that comes with SRI-CP. It is based on the following idea (Mertens 2005): 1) generate a random graph ensemble $G(n,p)$ according to the Erdos-Renyi model (Erdös and Rényi 1960), where n is the required number of agents and p is the edge probability (i.e., each pair of vertices is connected independently with probability p); 2) since the edges characterize the acceptability relations, generate a random permutation of each agent's acceptable partners to provide the preference lists. We define the *completeness degree* for an instance as the percentage $p * 100$.

In our experiments, we have used CLINGO (Version 5.2.2) on a machine with Intel Xeon(R) W-2155 3.30GHz CPU and 32GB RAM.

Scalability of SRTI-ASP*: SRI and its variations.* We have generated instances of different sizes, where the number of agents are 20, 40, 60, 80,100, 150 and 200, and the completeness degrees are 25%, 50%, 75% and 100%. For each number of agents and for each completeness degree, we have generated 20 instances. We have experimented with these randomly generated instances to analyze the scalability of SRTI-ASP for SRI, Egalitarian SRI, Rank Maximal SRI, and Almost SRI. The results are shown in Table 2. We make the following observations from this table:

O1 The computation times for finding a stable matching (if one exists) and finding out that there exists no stable matching are comparable to each other.

Consider, for instance, the completeness degree 25%, and 80 agents. For 13 (out of 20) instances, the average CPU time to compute a stable matching is 0.167 seconds. For the remaining 7 instances, the average CPU time to find that a stable matching does not exist is 0.183 seconds. These timings are comparable to each other.

O2 Computing an egalitarian stable matching generally takes slightly less time than computing a rank maximal stable matching.

For instance, for the 13 instances with a stable matching, computing egalitarian stable matchings takes on average 0.255 seconds; computing rank maximal stable matchings takes a similar amount of time, 0.256 seconds. For larger instances, we can observe that the latter takes a bit more time.

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

Table 2. *Scalability of* SRTI-ASP *in computation time, for SRI, Egalitarian (E) SRI, Rank Maximal (R) SRI, and Almost (A) SRI.*

		SRI				E SRI	R SRI	A SRI
completeness degree	$\|A\|$	#instances with a solution	average time (sec)	#instances without any solution	average time (sec)	average time (sec)	average time (sec)	average time (sec)
25%	40	11	0.029	9	0.024	0.204	0.201	0.017
	60	10	0.068	10	0.077	0.129	0.219	0.479
	80	13	0.167	7	0.183	0.255	0.256	4.416
	100	14	0.37	6	0.442	0.575	0.602	85.86
	150	8	1.995	12	1.994	14.126	17.703	TO
	200	10	8.794	10	8.323	59.18	83.85	TO
50%	40	11	0.06	9	0.084	0.106	0.108	0.136
	60	16	0.276	4	0.381	0.535	0.56	3.748
	80	13	0.852	7	1.106	1.82	1.85	3.748
	100	12	3.192	8	2.828	4.97	5.26	343.24
	150	14	15.880	6	15.59	153.54	149.447	TO
	200	9	69.65	11	72.08	524.58	704.3	TO
75%	40	14	0.136	6	0.207	0.722	0.33	0.486
	60	13	0.744	7	0.997	8.171	1.85	15.526
	80	8	3.971	12	3.346	6.99	7.16	144.80
	100	13	11.06	7	8.801	19.04	20.65	885.39
	150	9	50.520	11	51.700	492.7	529.16	TO
	200	12	175.81	8	202.46	1757.0	1475.0	TO
100%	40	15	0.227	5	0.301	0.463	0.56	0.675
	60	14	1.483	6	2.110	2.534	3.372	27.935
	80	13	7.472	7	7.426	14.39	15.47	181.09
	100	10	24.43	10	16.268	35.92	40.42	2627.77
	150	11	112.23	9	113.21	360.7	362.12	TO
	200	12	388.58	8	353.8	844.03	1147.0	TO

TO: Timeout (over 3000 seconds)

O3 Computing an almost stable matching significantly takes more time, compared to computing an egalitarian or a rank maximal stable matching.
O4 Solving the optimization variants of SRI takes significantly more time, compared to solving SRI.

For instance, for the 7 instances without any stable matching, computing almost stable matchings takes in 4.416 seconds on average.

The observations O2 and O3 are interesting, considering all the three optimization variations of SRI are NP-hard. Though, the observation O4 (better illustrated in Fig. 2) is not surprising, considering that SRI is in P (Table 1).

We can can further observe the following about scalability:

O5 As the completeness degree increases, the computation times increase.
O6 As the number of agents increases, the computation times increase.

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

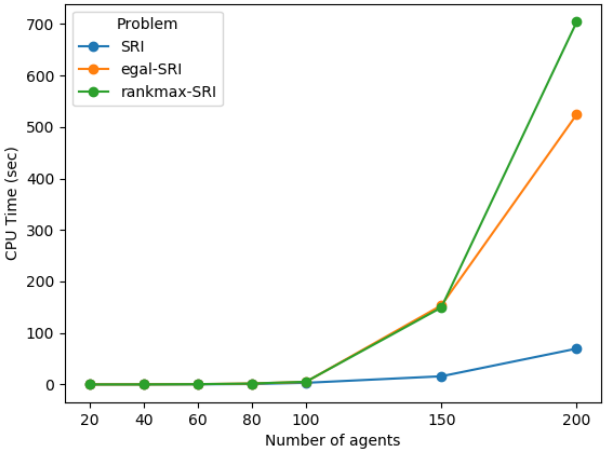

Fig. 2. Comparison of SRI with its optimization variants, when the completeness degree is 50%.

Table 3. *Scalability of* SRTI-ASP *(completeness degree of 25%): SRI vs. SRTI*

	SRI		SRTI							
			%25 tie		%50 tie		%75 tie		%100 tie	
$\|A\|$	#inst. with solns.	avg time (sec)	#inst. with solns.	avg time (sec)	#inst. with solns.	avg time (sec)	#inst. with solns.	avg time (sec)	#inst. with solns.	avg time (sec)
20	20	0.009	20	0.034	20	0.043	20	0.063	20	0.073
40	11	0.029	18	0.191	20	0.243	20	0.319	20	0.346
60	10	0.068	20	0.967	20	1.297	20	1.746	20	1.752
80	13	0.167	20	2.833	20	3.208	20	3.453	20	4.187
100	14	0.370	20	3.372	20	2.609	20	2.293	20	2.178

Scalability of SRTI-ASP*: SRI vs. SRTI.* To experiment with SRTI-ASP on SRTI instances (under weak stability), we have randomly generated ties for the randomly generated SRI instances with the completeness degree 25%. For each agent x, we have 1) identified the set T of agents that are not acceptable to x and vice versa, and randomly picked one of these agents, say y, 2) identified the set U of agents that are acceptable to x and vice versa, and randomly picked one of these agents, say z, and 3) added y in preference list of x so that x is indifferent between y and z. We have added ties as many as $25\%, 50\%, 75\%, 100\%$ of the number of agents. The results of these experiments are shown in Table 3. We can observe the following:

O7 Solving SRTI takes significantly more time, compared to solving SRI.
O8 SRI instances that do not have any stable matching, often have stable matchings after ties are added.

The observation O7 is expected, since SRTI is NP-complete whereas SRI is in P (Table 1). O8 is reasonable since adding ties reduces the number of potential blocking pairs in general, and thus allows SRTI-ASP to explore more possibilities.

Scalability of SRTI-ASP *vs.* SRI-CP. We have experimented with SRI-CP (SRItoolkit 2019), which utilizes the CP solver CHOCO (Version 2.1.5), on the SRI instances generated by SRI-CP's random instance generator. The results for 80–200 agents are shown in Table 4.

O9 For large SRI instances, SRI-CP performs significantly better than SRTI-ASP.

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

Table 4. SRTI-ASP *vs.* SRI-CP

completeness degree	#agents	#instances with a solution	SRTI-ASP average time (sec) exists solution	SRTI-ASP average time (sec) no solution	SRI-CP average time (sec) exists solution	SRI-CP average time (sec) no solution
25%	80	13	0.167	0.183	0.318	0.310
	100	14	0.370	0.442	0.389	0.492
	150	8	1.995	1.994	0.704	0.697
	200	10	8.794	8.323	1.047	0.999
50%	80	13	0.852	1.106	0.481	0.475
	100	12	3.192	2.828	0.674	0.646
	150	14	15.88	15.59	1.267	1.211
	200	9	69.65	72.08	1.940	1.960
75%	80	8	3.971	3.346	0.684	0.682
	100	13	11.062	8.801	0.967	0.961
	150	9	50.52	51.70	1.795	1.780
	200	12	175.81	202.46	3.214	3.263
100%	80	13	7.396	7.426	0.883	0.912
	100	10	24.435	16.268	1.190	1.175
	150	11	112.23	113.21	2.593	2.674
	200	12	388.58	353.8	5.074	5.061

This observation has led to the following idea (mentioned by Prosser (2014)) for SRI instances that have stable matchings: "Can we solve Egalitarian SRI faster than SRTI-ASP, by first enumerating all stable matchings using SRI-CP, and then finding the optimal one?" We have noticed that the instances (generated by the random instance generator of SRI-CP) generally have one or two stable matchings. In that case, the answer to this question is Yes. This observation contradicts with the theoretical result on the NP-hardness of Egalitarian SRI. So we have generated some instances with more stable matchings. For example, for an instance (sri90) with 90 agents and with more than 9 million stable matchings, SRTI-ASP takes 1.75 seconds to find an egalitarian stable matching whereas SRI-CP can not enumerate all these solutions (due to fast consumption of memory). For SRI instances with many stable matchings, it may be better to use SRTI-ASP to solve Egalitarian SRI; further investigations are planned as part of our future work.

Meanwhile, we have investigated a similar question for SRI instances that do not have any stable matching: "Can we solve Almost SRI instances with n agents faster than SRTI-ASP, by checking whether removing $\binom{n}{2}, \binom{n}{4}, ..$ agents (i.e., potential blocking pairs) leads to a stable matching?" We have observed that, for small SRI instances with one blocking pairs, the answer to this question is Yes: If we remove two agents, then we can find a stable matching. For larger instances with many blocking pairs, the answer is negative. For example, for an instance (sri60a) with 60 agents, that does not have any stable matching, we have observed that SRTI-ASP finds an almost stable matching with 10 blocking pairs in 9.057 seconds. With the enumerate-test method mentioned in the question, we have to enumerate $\binom{60}{2} + \binom{60}{4} + ... + \binom{60}{10}$ (more than 7×10^{10}) instances, and check them one by one using SRI-CP until an almost stable matching is found. Assuming that SRI-CP takes 0.001 seconds per instance, in the worst case we will have

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

Table 5. SRTI-ASP *vs.* SRI-AF

completeness degree	#agents	#instances with a solution	SRTI-ASP average time (sec) exists solution	SRTI-ASP average time (sec) no solution	SR-AF average time (sec) exists solution	SR-AF average time (sec) no solution
25%	80	13	0.167	0.183	0.118	0.126
	100	14	0.370	0.442	0.242	0.254
	150	8	1.995	1.994	1.002	1.077
	200	10	8.794	8.323	2.674	2.604
50%	80	13	0.852	1.106	0.521	0.527
	100	12	3.192	2.828	1.073	1.01
	150	14	15.88	15.59	5.029	4.918
	200	9	69.65	72.08	13.35	14.19
75%	80	8	3.971	3.346	1.248	1.252
	100	13	11.062	8.801	2.795	2.772
	150	9	50.52	51.70	12.65	12.222
	200	12	175.81	202.46	36.35	35.72
100%	80	13	7.396	7.426	2.525	2.523
	100	10	24.435	16.268	5.677	5.479
	150	11	112.23	113.21	24.94	24.75
	200	12	388.58	353.8	77.89	74.67

to test all 7×10^{10} instances, and it will take at least 2 years. This observation confirms with the theoretical result on the NP-hardness of Almost SRI; further investigations are planned as part of our future work.

Scalability of SRTI-ASP *vs.* SR-AF The ASP-based method SR-AF (Amendola 2018) utilizes abstract argumentation frameworks (Dung 1995) for the Stable Marriage problem. According to SR-AF, an argumentation framework AF=(Arg,Att) models an SR instance if the arguments in Arg are pairs of different agents, and the attacks $((a,b),(x,y))$ in $Att \subseteq Arg\times Arg$ satisfy the following properties: (i) $x = a$ and $b \prec_x y$, or (ii) $y = b$ and $a \prec_y x$. For every SR instance, once the arguments and attacks are generated, they are translated into a program P_{AF} using the existing methods (Wu et al. 2009). In particular, for every argument (a,b) in AF, if $(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)$ are the arguments that attack (a,b), the following rule is included in P_{AF}:

$$in(a,b) \leftarrow not\ in(x_1,y_1), not\ in(x_2,y_2),\ldots,not\ in(x_m,y_m).$$

There is a one-to-one correspondence between the answer sets for the program P_{AF} and the stable extensions of AF, due to Theorem 2 of Amendola (2018).

We have extended SR-AF's argumentation framework from SR to SRI, by defining the arguments as pairs of agents that are acceptable to each other. implemented (in Python) the transformation of an SRI instance into an argumentation framework AF and then to a program P_{AF}. We have experimented with this extended SR-AF (called SRI-AF from now on) on the SR instances generated by SRI-CP's random instance generator. The results are shown in Table 5.

O10 For large SRI instances, SRI-AF performs significantly better than SRTI-ASP.

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

6 Conclusion

We have developed a formal framework, called SRTI-ASP, that is sound and complete (Theorem 1) and general enough to provide solutions to various stable roommates problems, such as, SR, SRI, SRT, SRTI, Egalitarian SRTI, Rank Maximal SRTI, Almost SRTI. Except for SR and SRI, all these variations are intractable (Table 1). Our ongoing work involves extending SRTI-ASP to other computationally hard stable roommates problems.

Since SRTI-ASP utilizes Answer Set Programming (ASP), the formulations of problems are concise and elaboration tolerant, and thus SRTI-ASP provides a flexible framework to study variations of stable roommates problems. Having such a flexible framework and implementation is valuable for studies in matching theory.

We have evaluated SRTI-ASP over different sizes of randomly generated SRI instances, and have made many interesting observations (O1–O10) about its scalability over different intractable variations of SRI, and in comparison with SRI-CP and SR-AF over tractable variations of SR. The results of our empirical analysis of SRTI-ASP are promising, in particular, for computationally hard problems. Considering that the input sizes of the instances are large enough for many dormitories, the results of experiments are also promising for real-world applications.

Comparisons with SRI-CP and SRI-AF has helped us to better observe the flexibility of SRTI-ASP due to elaboration tolerant ASP representations. It is easier to extend SRTI-ASP to address different variations of SR, while SRI-CP and SRI-AF require further studies in modeling as well as implementation. Note that SRI-CP uses CHOCO via a Java wrapper, and SRI-AF solves SRI via an argumentation framework. As a future work, we plan to investigate how SRI-CP and SRI-AF can be extended to SRTI and its intractable versions.

The Stable Marriage problem with Ties (SMT) under strong stability, which can be solved with a polynomial time algorithm (Irving 1994), has been used as a benchmark in ASP competitions. Its representation is based on ranks instead of preferences, and does not utilize choice rules, cardinality expressions or weak constraints. With its intractable variations (under weak stability), SRTI-ASP contributes to ASP studies by providing an elaboration tolerant formulation and a complementary and rich set of benchmark instances of Stable Roommates problems.

References

ABRAHAM, D. J., BIRÓ, P., AND MANLOVE, D. F. 2005. "almost stable" matchings in the roommates problem. In *International Workshop on Approximation and Online Algorithms*. Springer, 1–14.

AMENDOLA, G. 2018. Solving the stable roommates problem using incoherent answer set programs. In *Proc. of RiCeRcA*.

ARKIN, E. M., BAE, S. W., EFRAT, A., OKAMOTO, K., MITCHELL, J. S. B., AND POLISHCHUK, V. 2009. Geometric stable roommates. *Inf. Process. Lett. 109,* 4, 219–224.

BIRÓ, P., MANLOVE, D. F., AND MCDERMID, E. J. 2012. "Almost stable" matchings in the roommates problem with bounded preference lists. *Theoretical Computer Science 432*, 10 – 20.

BREWKA, G., EITER, T., AND TRUSZCZYNSKI, M. 2016. Answer set programming: An introduction to the special issue. *AI Magazine 37,* 3, 5–6.

BUCCAFURRI, F., LEONE, N., AND RULLO, P. 2000. Enhancing disjunctive datalog by constraints. *IEEE Trans. Knowl. Data Eng. 12,* 5, 845–860.

COOPER, F. 2020. Fair and large stable matchings in the stable marriage and student-project allocation problems. Ph.D. thesis, University of Glasgow.

CSEH, Á., IRVING, R. W., AND MANLOVE, D. F. 2019. The stable roommates problem with short lists. *Theory of Computing Systems 63,* 1, 128–149.

https://doi.org/10.1017/S1471068420000277 Published online by Cambridge University Press

DANTSIN, E., EITER, T., GOTTLOB, G., AND VORONKOV, A. 2001. Complexity and expressive power of logic programming. *ACM Comput. Surv. 33,* 3, 374–425.

DUNG, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *AIJ 77,* 2, 321 – 357.

ERDEM, E. AND LIFSCHITZ, V. 2003. Tight logic programs. *TPLP 3,* 4-5, 499–518.

ERDOGAN, S. T. AND LIFSCHITZ, V. 2004. Definitions in answer set programming. In *Proc. of LPNMR*.

ERDÖS, P. AND RÉNYI, A. 1960. On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*. 17–61.

FEDER, T. 1992. A new fixed point approach for stable networks and stable marriages. *Journal of Computer and System Sciences 45,* 2, 233 – 284.

GAI, A.-T., MATHIEU, F., DE MONTGOLFIER, F., AND REYNIER, J. 2007. Stratification in P2P networks: Application to BitTorrent. In *Proc. of ICDCS*. 30.

GALE, D. AND SHAPLEY, L. S. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly 69,* 1, 9–15.

GEBSER, M., KAUFMANN, B., KAMINSKI, R., OSTROWSKI, M., SCHAUB, T., AND SCHNEIDER, M. T. 2011. Potassco: The Potsdam Answer Set Solving Collection. *AI Communications 24,* 2, 107–124.

GELFOND, M. AND LIFSCHITZ, V. 1988. The stable model semantics for logic programming. In *Proc. of ICLP*. 1070–1080.

GELFOND, M. AND LIFSCHITZ, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9*, 365–385.

GUSFIELD, D. AND IRVING, R. W. 1989. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, Cambridge, MA, USA.

IRVING, R. W. 1985. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms 6,* 4, 577–595.

IRVING, R. W. 1994. Stable marriage and indifference. *Discrete Applied Mathematics 48,* 3, 261–272.

IRVING, R. W. AND MANLOVE, D. F. 2002. The stable roommates problem with ties. *Journal of Algorithms 43,* 1, 85–105.

IRVING, R. W., MANLOVE, D. F., AND O'MALLEY, G. 2009. Stable marriage with ties and bounded length preference lists. *Journal of Discrete Algorithms 7,* 2, 213 – 219.

KNUTH, D. 1997. Stable marriage and its relation to other combinatorial problems: an introduction to the mathematical analysis of algorithms. In *Volume 10 of CRM Proc.* Mariages Stables (in English), 1976.

KUJANSUU, E., LINDBERG, T., AND MAKINEN, E. 1999. The stable roomates problem and chess tournament pairings. *Divulgaciones Matemáticas 7*, 19–28.

KUNYSZ, A. 2016. The Strongly Stable Roommates Problem. In *Proc. of ESA*. Vol. 57. 60:1–60:15.

LIFSCHITZ, V. AND TURNER, H. 1994. Splitting a logic program. In *Proc. of ICLP*. 23–37.

MANLOVE, D. 1999. Stable marriage with ties and unacceptable partners.

MATCHINGTOOLKIT. 2020. `http://matwa.optimalmatching.com/`.

MCCARTHY, J. 1998. Elaboration tolerance. In *Proc. of CommonSense*.

MERTENS, S. 2005. Random stable matchings. *Journal of Statistical Mechanics: Theory and Experiment 2005,* 10, P10008–P10008.

PROSSER, P. 2014. Stable roommates and constraint programming. In *Proc. of CPAIOR*. 15–28.

RONN, E. 1990. Np-complete stable matching problems. *Journal of Algorithms 11,* 2, 285 – 304.

ROTH, A. E., SÖNMEZ, T., AND ÜNVER, M. U. 2005. Pairwise kidney exchange. *Journal of Economic Theory 125,* 2, 151 – 188.

SCOTT, S. 2005. A study of stable marriage problems with ties. Ph.D. thesis, University of Glasgow.

SIMONS, P., NIEMELAE, I., AND SOININEN, T. 2002. Extending and implementing the stable model semantics. *AIJ 138,* 1, 181–234.

SRITOOLKIT. 2019. `http://www.dcs.gla.ac.uk/~pat/roommates/distribution/` (2019-11-21).

WU, Y., CAMINADA, M., AND GABBAY, D. M. 2009. Complete extensions in argumentation coincide with 3-valued stable models in logic programming. *Studia Logica 93,* 2-3, 383–403.