CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	A collection of constraint programming models for the three-dimensional stable matching problem with cyclic preferences
Authors	Cseh, Ágnes;Escamocher, Guillaume;Genç, Begüm;Quesada, Luis
Publication date	2021-10-15
Original Citation	Cseh, Á, Escamocher, G., Genç, B. and Quesada, L. (2021) 'A collection of Constraint Programming models for the three-dimensional stable matching problem with cyclic preferences', 27th International Conference on Principles and Practice of Constraint Programming (CP 2021), Montpellier, France, 25 - 29 October, 22 (19pp). doi: 10.4230/LIPIcs.CP.2021.22
Type of publication	Conference item
Link to publisher's version	https://doi.org/10.5281/zenodo.5156119 - 10.4230/LIPIcs.CP.2021.22
Rights	© 2021, Ágnes Cseh, Guillaume Escamocher, Begüm Genç, and Luis Quesada. Licensed under Creative Commons License CC-BY 4.0 - https://creativecommons.org/licenses/by/4.0/
Download date	2026-11-01 16:07:06
Item downloaded from	https://hdl.handle.net/10468/12115

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

A Collection of Constraint Programming Models for the Three-Dimensional Stable Matching Problem with Cyclic Preferences

Ágnes Cseh
Hasso-Plattner-Institute, Universität Potsdam, Germany
Institute of Economics, Centre for Economic and Regional Studies, Pécs, Hungary

Guillaume Escamocher
Insight Centre for Data Analytics, School of Computer Science and Information Technology, University College Cork, Ireland

Begüm Genç
Insight Centre for Data Analytics, School of Computer Science and Information Technology, University College Cork, Ireland

Luis Quesada
Insight Centre for Data Analytics, School of Computer Science and Information Technology, University College Cork, Ireland

Abstract

We introduce five constraint models for the 3-dimensional stable matching problem with cyclic preferences and study their relative performances under diverse configurations. While several constraint models have been proposed for variants of the two-dimensional stable matching problem, we are the first to present constraint models for a higher number of dimensions. We show for all five models how to capture two different stability notions, namely weak and strong stability. Additionally, we translate some well-known fairness notions (i.e. sex-equal, minimum regret, egalitarian) into 3-dimensional matchings, and present how to capture them in each model.

Our tests cover dozens of problem sizes and four different instance generation methods. We explore two levels of commitment in our models: one where we have an individual variable for each agent (individual commitment), and another one where the determination of a variable involves pairing the three agents at once (group commitment). Our experiments show that the suitability of the commitment depends on the type of stability we are dealing with. Our experiments not only led us to discover dependencies between the type of stability and the instance generation method, but also brought light to the role that learning and restarts can play in solving this kind of problems.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Theory of computation → Design and analysis of algorithms

Keywords and phrases Three-dimensional stable matching with cyclic preferences, 3DSM-cyc, Constraint Programming, fairness

Digital Object Identifier 10.4230/LIPIcs.CP.2021.22

Supplementary Material *Dataset*: `https://doi.org/10.5281/zenodo.5156119`

Funding This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant numbers 12/RC/2289-P2, 16/SP/3804 and 16/RC/3918, which are co-funded under the European Regional Development Fund.
Ágnes Cseh: The Federal Ministry of Education and Research of Germany in the framework of KI-LAB-ITSE (project number 01IS19066), OTKA grant K128611, János Bolyai Research Fellowship.

Acknowledgements COST Action CA16228 European Network for Game Theory.



© Ágnes Cseh, Guillaume Escamocher, Begüm Genç, and Luis Quesada;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

In the classic stable marriage problem, we are given a bipartite graph, where the two sets of vertices represent men and women, respectively. Each vertex has a strictly ordered preference list over his or her possible partners. A matching is *stable* if it is not *blocked* by any edge, that is, no man-woman pair exists who are mutually inclined to abandon their partners and marry each other. Stable matchings were first formally defined in the seminal paper of Gale and Shapley [20], who introduced the terminology based on marriage that since then became wide-spread. The notion was then extended to non-bipartite graphs by Irving [27]. Variants of stable matching problems are widely used in employer allocation markets [44], university admission decisions [3, 7], campus housing assignments [8, 42] and bandwidth allocation [19]. Typically, the aim is to solve the decision problem on whether a stable matching exists, or even to solve an optimisation problem considering different fairness notions among stable matchings, such as egalitarian, minimum-regret, or sex-equal.

A natural generalisation of the problem, as suggested by Knuth in his influential book [31], is to extend the two-sided stable marriage problem to three sets of agents. Two input variants of this extension have been defined in the literature. In the first variant, called the *3-gender stable marriage problem* (3GSM) problem [2, 38], each agent has a preference list over the n^2 pairs of agents from the other two sets, assuming that each agent set contains n agents. Another way of generalising stable matching to three agent sets is the *3-dimensional stable matching problem with cyclic preferences* (3DSM-CYC) [38], in which agents from the first set only have preferences over agents from the second set, agents from the second set only have preferences over agents from the third set, and agents from the third set only have preferences over agents from the first set. In both problem variants, the aim is to find a matching that does not admit a blocking triple, where a blocking triple can have slightly different definitions depending on whether the preference lists contain ties or whether a strict improvement for all agents is required. We explore these different notions in Section 1.1.

1.1 3-dimensional stable matching

In the 3GSM problem variant, the default stability notion is called weak stability, according to which a blocking triple is defined as a set of three agents, all of whom would strictly improve their current match if they would form a triple in the solution. Deciding whether a stable matching exists in a given instance is NP-complete even if the preference lists are complete [38, 47]. A highly restricted preference structure was later identified that allows for a polynomial-time algorithm for the same decision problem [12]. Research then evolved in the direction of preference lists with ties, which gives rise to four different stability definitions, namely weak, strong, super, and ultra stability, and in the direction of consistent preferences, which is a naturally restricted preference domain [26].

The derived research results appear to be more diverse when it comes to the 3DSM-CYC problem variant. Firstly, two stability notions have been investigated: weak and strong. A *weakly stable matching* does not admit a blocking triple such that all three agents would improve, while according to *strong stability*, a triple already blocks if at least one of its agents improves, and the others in the triple remain equally satisfied. Biró and McDermid [5] showed that deciding whether a weakly stable matching exists is NP-complete if preference lists are allowed to be incomplete, and that the same complexity result holds for strong stability even with complete lists. However, the combination of complete lists and weak stability proved to be extremely challenging to solve.

For this setting, Boros et al. [6] proved that each 3DSM-CYC instance admits a weakly stable matching for $n \leq 3$, where n is the size of each vertex set in the tripartition. Eriksson et al. [15] later extended this result to $n \leq 4$. Additionally, Pashkovich and Poirrier [41] further proved that not only one, but at least two stable matchings exist for each instance with $n = 5$. By this time, the conjecture on the guaranteed existence of a weakly stable matching in 3DSM-CYC with complete lists became one of the most riveting open questions in the matching under preferences literature [31, 33, 50]. Surprisingly, Lam and Plaxton [32] recently disproved this conjecture by showing that weakly stable matchings for 3DSM-CYC need not exist for an arbitrary n, moreover, it is NP-complete to determine whether a given 3DSM-CYC instance with complete lists admits a weakly stable matching.

Application-oriented research has focused on the so-called "3-sided matching with cyclic and size preferences" problem, defined by Cui and Jia [11]. They modeled three-sided networking services, such as frameworks connecting users, data sources, and servers. In their setting, users have identical preferences over data sources, data sources have preferences over servers based on the transferred data, and servers have preferences over users. The characterising feature of this variant is that a triple might contain more than one user, as servers aim at maximizing the number of users assigned to them. This feature clearly differentiates the problem from the classic 3DSM-CYC setting. Building upon this work, Panchal and Sharma [40] provided a distributed algorithm that finds a stable solution. Raveendran et al. [43] tested resource allocation in Network Function Virtualisation. They demonstrated the superior performance of the proposed cyclic stable matching framework in terms of data rates and user satisfaction, compared to a centralised random allocation approach.

1.2 Constraint Programming approaches for finding stable matchings

Gent et al. [22] were the first to propose Constraint Programming (CP) models for the classic stable marriage problem. They showed that it is possible to obtain man-optimal and woman-optimal stable matchings immediately from the solution by enforcing Arc Consistency (AC). Later, Unsworth and Prosser [48, 49] presented a binary constraint for the the same problem and showed that their encoding is better in terms of space and time when compared to Gent et al.'s approach. They also investigated sex-equal stable matchings in their studies.

The next milestone was reached by Manlove et al. [34], who proposed three CP models for the Hospital / Residents problem (HR), which is the many-to-one generalisation of the stable marriage problem. They also explored side constraints for their models such as the case with forbidden pairs, residents who may form groups, or residents who may swap their hospitals. The existing research shows that CP models for the stable marriage problem with incomplete lists and for HR are tractable [22, 34]. O'Malley further explored CP models in his thesis for the stable marriage problem, and presented four constraint models [39]. Later on, Siala and O'Sullivan [45] improved the cloned model of Manlove et al. [34] by using a global constraint that achieves Bound Consistency in linear time.

In 2012 Eirinakis et al. [14] used the poset graph of rotations to enumerate all solutions of HR, and presented an improved version to the direct CP model of Manlove et al. [34]. Subsequently, Siala and O'Sullivan [46] used the rotation poset to model stable matchings as SAT formulation for all three types of problems: one-to-one, one-to-many, and many-to-many. They presented empirical results for finding sex-equal stable matchings, and showed that their approach outperforms the model presented in their previous paper [45]. Additionally, Drummond et al. [13] used SAT encoding for finding stable matchings that include couples.

To the best of our knowledge, CP or SAT models for 3-dimensional stable matchings have not been studied before.

2 Preliminaries

In this section we introduce the terminology and notation for the problem variants we will study. First we formalise the 3DSM-CYC problem and define the two known stability concepts for it. Then, we define three standard fairness notions that were constructed to distinguish balanced stable solutions on bipartite and non-bipartite stable matching instances.

2.1 3-dimensional stable matching with cyclic preferences

Input and output. Formally, a 3DSM-CYC instance is defined over three disjoint sets of agents of size n, denoted by $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$, and $C = \{c_1, \ldots, c_n\}$. A *matching* M corresponds to a disjoint set of triples, where each triple, denoted by (a_i, b_j, c_k), contains exactly one agent from each agent set. Each agent is equipped with her own preferences in the input. The cyclic property of the preferences means the following: each agent in A has a strict and complete preference list over the agents in B, each agent in B has a strict and complete preference list over the agents in C, and finally, each agent in C has a strict and complete preference list over the agents in A. These preferences are captured by the *rank function*, where $\text{rank}_{a_i}(b_j)$ is the position of agent b_j in the preference list of a_i, from 1 if b_j is a_i's most preferred agent to n if b_j is a_i's least preferred agent.

Preferences over triples. The preference relation of an agent on possible triples can be derived naturally from the preference list of this agent. Agent a_i is indifferent between triples (a_i, b_j, c_{k_1}) and (a_i, b_j, c_{k_2}), since she only has preferences over the agents in B and the same agent b_j appears in both triples. However, when comparing triples (a_i, b_{j_1}, c_{k_1}) and (a_i, b_{j_2}, c_{k_2}), where $b_{j_1} \neq b_{j_2}$, a_i prefers the first triple if $\text{rank}_{a_i}(b_{j_1}) < \text{rank}_{a_i}(b_{j_2})$, and she prefers the second triple otherwise. The preference relation is defined analogously for agents in B and C as well.

Weak and strong stability. A triple $t = (a_i, b_j, c_k)$ is said to be a *strongly blocking triple* to matching M if each of a_i, b_j, and c_k prefer t to their respective triples in M. Practically, this means that a_i, b_j, and c_k could abandon their triples to form triple t on their own, and each of them would be strictly better off in t than in M. If a matching M does not admit any strongly blocking triple, then M is called a *weakly stable* matching. Similarly, a triple $t = (a_i, b_j, c_k)$ is called a *weakly blocking triple* if at least two agents in the triple prefer t to their triple in M, while the third agent does not prefer her triple in M to t. This means that at least two agents in the triple can improve their situation by switching to t, while the third agent does not mind the change. A matching that does not admit any weakly blocking triple is referred as *strongly stable*. By definition, strongly stable matchings are also weakly stable, but not the other way round. Observe that it is impossible to construct a triple t that keeps exactly two agents of a triple equally satisfied, while making the third agent happier, since the earlier two agents need to keep their partners to reach this, which then already defines the triple as one already in M.

2.2 Fair stable solutions

In this paper, we translate some standard fairness notions from the classic stable marriage problem to 3DSM-CYC. In most stable matching problems, several stable solutions might be present, which gives way to choosing a fair or balanced one among them. We now review the most prevalent fairness notions in such decisions [25, 29, 33, 10].

Egalitarian stable matchings. Possibly the most natural way to define a good stable matching is captured by the notion of *egalitarian stable matchings*. Each agent's satisfaction can be measured by how high she ranks her partner in the matching. In order to gain a comprehensive measure, we sum up those ranks for all matched agents. A stable matching is called egalitarian, if it minimises this sum among all stable matchings. Finding an egalitarian stable matching in the classic stable marriage problem can be done in polynomial time [28, 24], while it is NP-hard, but 2-approximable if the underlying instance is non-bipartite [17, 18]. An egalitarian stable matching in a 3DSM-CYC instance is defined as the extreme point of the following function.

$$\min_{M \text{ is a stable matching}} \left\{ \sum_{(a_i,b_j,c_k)\in M} \text{rank}_{a_i}(b_j) + \text{rank}_{b_j}(c_k) + \text{rank}_{c_k}(a_i) \right\} \tag{1}$$

Minimum regret stable matchings. Another popular fairness notion, called *minimum regret stable matching*, intuitively maximises the satisfaction of the least satisfied person in the instance. In this context, each agent's regret is measured by how high she ranks her partner in the matching – the larger this rank is, the more regret she experiences. The regret of matching M is defined as the largest regret in the instance, i.e. the worst rank that appears in the matching. Finding a minimum regret stable matching can be done in polynomial time both in bipartite and non-bipartite instances [23, 24]. A minimum regret stable matching in a 3DSM-CYC instance is defined as the extreme point of the following function.

$$\min_{M \text{ is a stable matching}} \left\{ \max_{(a_i,b_j,c_k)\in M} \left\{ \text{rank}_{a_i}(b_j), \text{rank}_{b_j}(c_k), \text{rank}_{c_k}(a_i) \right\} \right\} \tag{2}$$

Sex-equal stable matchings. A third condition is called *sex-equality*, which aims at reaching the same satisfaction level of each agent set. The satisfaction of a set of agents is measured by summing up the satisfaction level, that is, the rank of the matching partner, of each agent in the set. In the classic stable marriage setting, a sex-equal stable matching minimises the difference between the satisfaction level of the two sets. Finding a sex-equal stable matching is NP-hard in those instances [30, 35]. Even though the notion cannot be defined for non-bipartite instances, it translates readily to 3DSM-CYC instances. The difference of satisfaction level between any two of the three agent sets can be computed exactly as in the classic stable marriage setting. Then, the sum of the three pairwise differences must be minimised. We define a sex-equal stable matching as the extreme point of the following function.

$$\begin{aligned} \min_{M \text{ is a stable matching}} \Bigg\{ & \left| \sum_{(a_i,b_j,c_k)\in M} \text{rank}_{a_i}(b_j) - \text{rank}_{b_j}(c_k) \right| + \\ & \left| \sum_{(a_i,b_j,c_k)\in M} \text{rank}_{b_j}(c_k) - \text{rank}_{c_k}(a_i) \right| + \\ & \left| \sum_{(a_i,b_j,c_k)\in M} \text{rank}_{c_k}(a_i) - \text{rank}_{a_i}(b_j) \right| \Bigg\} \end{aligned} \tag{3}$$

2.3 Our contribution

This paper is the first to model 3-dimensional stable matchings, specifically the 3DSM-CYC problem, including its optimisation variants using side constraints. We propose the following CP models to find a stable matching in the 3DSM-CYC problem: DIV (divided agent sets), UNI (unified agent sets), and HS (hitting set). We implement each one of the models under both weak and strong stability. For the DIV and UNI models, we investigate two kinds of domain values: one based on the unique identifiers of the agents themselves, referred as *agent-based (agents)*, and the other based on the ranks of agents in one's preference list, referred as *rank-based (ranks)*. We first use the models to find any satisfying solution to a given 3DSM-CYC instance. Subsequently, we extend all models to optimisation variants under different fairness criteria and conclude with some empirical findings.

3 Methodology

In this section we present the details of our five proposed models. For each model, we propose the mandatory matching and stability constraints. We also propose how to model the fairness constraints for different optimisation versions. Furthermore, if we identified any, we state the redundant constraints that help the models with better pruning the search space.

3.1 Agent-based DIV model

The DIV-agents model consists of $3n$ variables $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$, and $Z = \{z_1, \ldots, z_n\}$, where the domain of each variable v is set as $D(v) = \{1, \ldots, n\}$. For agent-based domain values, assigning $x_i = j$ (respectively $y_i = j$, or $z_i = j$) corresponds to matching a_i to b_j (respectively b_i to c_j, or c_i to a_j). A stable matching M, if any exists, is found by using the following constraints.

- (matching) For all $1 \leq i, j, k \leq n$, we add the constraint $x_i = j \wedge y_j = k \Rightarrow z_k = i$. This is to ensure that each solution corresponds to a feasible, if not stable, matching.
- (stability) Under *weak* stability, for all $1 \leq i, j, k \leq n$, and for all i', j', k' such that a_i prefers b_j to $b_{j'}$, b_j prefers c_k to $c_{k'}$ and c_k prefers a_i to $a_{i'}$ we add the constraint $x_i \neq j' \vee y_j \neq k' \vee z_k \neq i'$. This is to ensure that there is no strongly blocking triple. When solving the problem under *strong* stability, the condition to post the constraint becomes: a_i prefers b_j to $b_{j'}$ or $j' = j$, and b_j prefers c_k to $c_{k'}$ or $k' = k$, and c_k prefers a_i to $a_{i'}$ or $i' = i$, and $i' \neq i \vee j' \neq j \vee k' \neq k$. Here, as well as in the other models, the difference between weak and strong stability constraints is that the latter also cover the case when exactly two agents of a potential blocking triple are matched together.
- (redundancy) For all $1 \leq i, j, k \leq n$, we add the constraint $y_j = k \wedge z_k = i \Rightarrow x_i = j$.
- (redundancy) For all $1 \leq i, j, k \leq n$, we add the constraint $z_k = i \wedge x_i = j \Rightarrow y_j = k$.
- (redundancy) We add AllDifferent(X) and AllDifferent(Y) and AllDifferent(Z) to ensure each agent has exactly one partner from each set.
- (optimisation) When solving a fair version of the problem, we add a constraint to minimise the objective in one of the following ways, depending on which notion of fairness is desired:
  - For egalitarian M, we model Eqn. 1 as: $\sum(\text{rank}_{a_i}(b_j) + \text{rank}_{b_j}(c_k) + \text{rank}\, c_k(a_i))$ for all i, j, k such that $x_i = j \wedge y_j = k \wedge z_k = i$.
  - For minimum regret M, we model Eqn. 2 as: $\max(\max(\text{rank}_{a_i}(b_j), \text{rank}_{b_j}(c_k), \text{rank}\, c_k(a_i)))$ for all i, j, k such that $x_i = j \wedge y_j = k \wedge z_k = i$.

- For sex-equal M, we model Eqn. 3 as: $\left|S_A - S_B\right| + \left|S_B - S_C\right| + \left|S_C - S_A\right|$ where $S_A = \sum(\text{rank}_{a_i}(b_j))$ for all i, j such that $x_i = j$, $S_B = \sum(\text{rank}_{b_j}(c_k))$ for all j, k such that $y_j = k$, and $S_C = \sum(\text{rank}_{c_k}(a_i))$ for all k, i such that $z_k = i$.

3.2 Rank-based DIV model

Variables and domains are the same in the rank-based DIV model (DIV-ranks) as they are in the agent-based DIV model (DIV-agents), but this time assigning $x_i = j$ (respectively $y_i = j$, or $z_i = j$) corresponds to matching a_i to her j^{th} preferred agent (respectively b_i to her j^{th} preferred agent, or c_i to her j^{th} preferred agent), who might be different from b_j. A stable matching M, if any exists, is found by using the following constraints.

- (matching) For all $1 \leq i, j, k \leq n$, we add the constraint $x_i = \text{rank}_{a_i}(b_j) \wedge y_j = \text{rank}_{b_j}(c_k) \Rightarrow z_k = \text{rank}_{c_k}(a_i)$. This is to ensure that each solution corresponds to a feasible, if not stable, matching.
- (stability) Under *weak* stability, for all $1 \leq i, j, k \leq n$, we add the constraint $x_i \leq \text{rank}_{a_i}(b_j) \vee y_j \leq \text{rank}_{b_j}(c_k) \vee z_k \leq \text{rank}_{c_k}(a_i)$. This is to ensure that there is no strongly blocking triple. When solving the problem under *strong* stability, the inequalities are strict but the following part is added to each disjunction: $\vee(x_i = \text{rank}_{a_i}(b_j) \wedge y_j = \text{rank}_{b_j}(c_k) \wedge z_k = \text{rank}_{c_k}(a_i))$.
- (redundancy) For all $1 \leq i, j, k \leq n$, we add the constraint $y_j = \text{rank}_{b_j}(c_k) \wedge z_k = \text{rank}_{c_k}(a_i) \Rightarrow x_k = \text{rank}_{a_i}(b_j)$.
- (redundancy) For all $1 \leq i, j, k \leq n$, we add the constraint $z_k = \text{rank}_{c_k}(a_i) \wedge x_i = \text{rank}_{a_i}(b_j) \Rightarrow y_j = \text{rank}_{b_j}(c_k)$.
- (optimisation) We add a constraint to minimise the objective in one of the following ways, depending on which notion of fairness is desired:
  - For egalitarian M, we model Eqn. 1 as: $\sum_{i=1}^{n}(x_i + y_i + z_i)$.
  - For minimum regret M, we model Eqn. 2 as: $\max(\max(x_i, y_i, z_i))$ for all $1 \leq i \leq n$.
  - For sex-equal M, we model Eqn. 3 as: $\left|S_A - S_B\right| + \left|S_B - S_C\right| + \left|S_C - S_A\right|$ where $S_A = \sum_{i=1}^{n}(x_i)$, $S_B = \sum_{j=1}^{n}(y_j)$, and $S_C = \sum_{k=1}^{n}(z_k)$.

Note that, as opposed to agent-based domains, there are no AllDifferent constraints in rank-based models. The reason for this is that with rank-based domains it is possible for two agents in the same agent set to be assigned the same value, for example if they both got assigned to their most preferred agent.

3.3 Agent-based UNI model

The UNI-agents model consists of n variables $X = \{x_1, \ldots, x_n\}$, where the domain of each variable v is set as $D(v) = \{(1,1), \ldots, (1,n), (2,1), \ldots, (n,n)\}$. Each tuple domain variable is implemented as an integer domain variable by representing the tuple (j,k) with the integer $(j-1)n + k$. For agent-based domain values, assigning (j,k) to x_i corresponds to having the triple (a_i, b_j, c_k) in the matching. A stable matching M, if any exists, is found by using the following constraints.

In both the current and following subsections, we denote by $x_{i,B}$ and $x_{i,C}$ respectively the first and second elements of the pair assigned to x_i.

- (matching) For all $1 \leq i < i' \leq n$, we add the constraint $x_{i,B} \neq x_{i',B} \wedge x_{i,C} \neq x_{i',C}$. This is to ensure that each solution corresponds to a feasible, if not stable, matching.

- (stability) Under *weak* stability, for all $1 \leq i, j, k \leq n$, for all $1 \leq i', i'', j', k' \leq n$ such that a_i prefers b_j to $b_{j'}$, b_j prefers c_k to $c_{k'}$ and c_k prefers a_i to $a_{i'}$, we add the constraint $(x_{i,B} \neq j') \vee (x_{i''} \neq (j, k')) \vee (x_{i',C} \neq k)$. This is to ensure that no triple is blocking. Because in UNI only the agents from A have their own associated variables, determining whether b_j was assigned to $c_{k'}$ requires checking for each agent $a_{i''}$ from A whether both b_j and $c_{k'}$ were assigned to $a_{i''}$. This is the reason for the additional index i''. When solving the problem under *strong* stability, the condition to post the constraint becomes: a_i prefers b_j to $b_{j'}$ or $j' = j$, and b_j prefers c_k to $c_{k'}$ or $k' = k$, and c_k prefers a_i to $a_{i'}$ or $i' = i$, and $i' \neq i \vee j' \neq j \vee k' \neq k$.
- (redundancy) We impose the constraint AllDifferent(X).
- (redundancy) Denote by $F(i)$ the set of tuples that have an agent i as their first element, and $S(i)$ the tuples that have i as their second. Then, for all agents $\forall i \in n$ we have $\sum_{j \in F(i)} \text{Count}(j, X) = 1$ and $\sum_{j \in S(i)} \text{Count}(j, X) = 1$.
- (optimisation) We add a constraint to minimise the objective in one of the following ways, depending on which notion of fairness is desired:
  - For egalitarian M, we model Eqn. 1 as: $\sum_{i=1}^{n}(\text{rank}_{a_i}(b_{x_{i,B}}) + \text{rank}_{b_{x_{i,B}}}(c_{x_{i,C}}) + \text{rank}_{c_{x_{i,C}}}(a_i))$.
  - For minimum regret M, we model Eqn. 2 as: $\max(\max(\text{rank}_{a_i}(b_{x_{i,B}}), \text{rank}_{b_{x_{i,B}}}(c_{x_{i,C}}), \text{rank}_{c_{x_{i,C}}}(a_i)))$ for all $1 \leq i \leq n$.
  - For sex-equal M, we model Eqn. 3 as: $\left|S_A - S_B\right| + \left|S_B - S_C\right| + \left|S_C - S_A\right|$ where $S_A = \sum_{i=1}^{n}(\text{rank}_{a_i}(b_{x_{i,B}}))$, $S_B = \sum_{i=1}^{n}(\text{rank}_{b_{x_{i,B}}}(c_{x_{i,C}}))$, and $S_C = \sum_{i=1}^{n}(\text{rank}_{c_{x_{i,C}}}(a_i))$.

3.4 Rank-based UNI model

Variables and domains are implemented the same in the rank-based UNI model (UNI-ranks) as they are in the agent-based UNI model (UNI-agents), but this time assigning (j, k) to x_i corresponds to matching a_i to her j^{th} preferred agent from B, and matching the latter to her k^{th} preferred agent from C. A stable matching M, if any exists, is found by using the following constraints.

- (matching) For all $1 \leq i < i' \leq n$, we add the constraint $\text{pref}_{a_i}(x_{i,B}) \neq \text{pref}_{a_{i'}}(x_{i',B}) \wedge \text{pref}_{\text{pref}_{a_i}(x_{i,B})}(x_{i,C}) \neq \text{pref}_{\text{pref}_{a_{i'}}(x_{i',B})}(x_{i',C})$, where $\text{pref}_{a_i}(r)$ (respectively $\text{pref}_{b_j}(r)$, or $\text{pref}_{c_k}(r)$) represents the agent $b \in B$ (respectively $c \in C$, or $a \in A$) such that $\text{rank}_{a_i}(b) = r$ (respectively $\text{rank}_{b_j}(c) = r$, or $\text{rank}_{c_k}(a) = r$). This is to ensure that each solution corresponds to a feasible, if not stable, matching.
- (stability) Under *weak* stability for all $1 \leq i, j, k \leq n$, for all $1 \leq i', i'', j'' \leq n$ such that c_k strictly prefers a_i to $a_{i'}$, we add the constraint $(x_{i,B} \leq \text{rank}_{a_i}(b_j)) \vee (x_{i'',B} \neq \text{rank}_{a_{i''}}(b_j)) \vee (x_{i'',C} \leq \text{rank}_{b_j}(c_k)) \vee (x_{i'} \neq (\text{rank}_{a_{i'}}(b_{j''}), \text{rank}_{b_{j''}}(c_k)))$. This is to ensure that no triple is blocking. When solving the problem under *strong* stability, c_k's preference of a_i to $a_{i'}$ is not strict (i' can be equal to i) but the two inequalities are, and to each disjunction is added the following part: $\vee(x_i = (\text{rank}_{a_i}(b_j), \text{rank}_{b_j}(c_k)))$.
- (optimisation) We add a constraint to minimise the objective in one of the following ways, depending on which notion of fairness is desired:
  - For egalitarian M, we model Eqn. 1 as: $\sum_{i=1}^{n}(x_{i,B} + x_{i,C} + \text{rank}_{\text{pref}_{\text{pref}_{a_i}(x_{i,B})}(x_{i,C})}(a_i))$.
  - For minimum regret M, we model Eqn. 2 as: $\max(\max(x_{i,B}, x_{i,C}, \text{rank}_{\text{pref}_{\text{pref}_{a_i}(x_{i,B})}(x_{i,C})}(a_i)))$ for all $1 \leq i \leq n$.
  - For sex-equal M, we model Eqn. 3 as: $\left|S_A - S_B\right| + \left|S_B - S_C\right| + \left|S_C - S_A\right|$ where $S_A = \sum_{i=1}^{n}(x_{i,B})$, $S_B = \sum_{i=1}^{n}(x_{i,C})$, and $S_C = \sum_{i=1}^{n}(\text{rank}_{\text{pref}_{\text{pref}_{a_i}(x_{i,B})}(x_{i,C})}(a_i))$.

3.5 HS model

In the HS model, let T be the set of all possible triples as $\{(1,1,1),(1,1,2),\ldots,(n,n,n)\}$. Without loss of generality, assume that the triples in T are ordered, so $t_i \in T$ refers to the i^{th} triple of T. Given a triple $t \in T$, we denote by $BT(t)$ all the triples in T that prevent t from becoming a blocking triple given the preferences. Then, finding a stable matching is equivalent to finding a hitting set of the non-blocking triples in T.

- Let M be a set variable whose upper bound is $\{i : t_i \in T\}$.
- (matching) Ensure that each agent from each set is matched by having:
  - $\forall a \in A : \sum_{t_i \in T : a \in t_i} (i \in M) = 1$;
  - $\forall b \in B : \sum_{t_i \in T : b \in t_i} (i \in M) = 1$;
  - $\forall c \in C : \sum_{t_i \in T : c \in t_i} (i \in M) = 1$.
- (stability) The stable matching is a hitting set of the non-blocking triples: $\forall t_j \in T : M \cap \{i : t_i \in BT(t_j)\} \neq \emptyset$. The type of stability is addressed in the computation of the BT sets. The model as such is not concerned with this aspect.

In this model, M is constrained to be a set of triples representing the stable matching as defined in Section 2.2, so egalitarian M, minimum regret M, and sex-equal M are defined as in Equations 1, 2, and 3 respectively.

In the actual implementation, M is represented in terms of an array of n^3 Boolean variables, where each variable refers to the inclusion/exclusion of the corresponding tuple in the mapping.

4 Experiments

We performed our experiments on machines with Intel(R) Xeon(R) CPU with 2.40GHz running on Ubuntu 18.04. Our initial experiments on small instances comparing all models are performed using Gecode 6.3.0 [21]. Then, we conduct further experiments by using our best performing models for larger instances on a constraint solver based on lazy-clause generation, namely Chuffed 0.10.4. [9]. For DIV and UNI models, instances were first processed by MiniZinc 2.5.5 [37] before being given to the solvers. The HS model has been directly encoded using Gecode 6.2.0. In Section 4.1 we describe the datasets in use. Then, in Sections 4.2 and 4.3 we compare the proposed models.

4.1 Dataset description

For every size n present in our experiments, we generated 100 instances with n agents in each agent set and a complete list for each agent. Half of these instances are random and the other half have some or all of the preferences based on master lists. Master list instances are instances where the preference lists of all agents in the same agent set are identical. Master lists provide a natural way to represent the fact that in practice agent preferences are often not independent. Examples of real-life applications of master lists occur in resident matching programs [4], dormitory room assignments [42], cooperative download applications such as BitTorrent [1], and 3-sided networking services [11]. The detailed distribution of the 100 instances generated for each size is as follows:

- **Random**: 50 random instances from uniform distribution.
- **ML_oneset**: 20 instances where the preference lists of the agents in one of the agent sets are based on master lists, and the preference lists of the agents in the other two agent sets are random.

- **ML_1swap**: 15 instances, where each agent set has a randomly chosen master list that all agents in the set follow. Then, we randomly choose two agents from each agent's preference list, and swap their positions.
- **ML_2swaps**: 15 instances, where each agent set has a randomly chosen master list that all agents in the set follow. First, we randomly choose two agents from each agent's preference list, and swap their positions. Subsequently, we randomly choose two more agents from each list such that the new agents were not involved in the first swap, then we swap their positions.

Note that neither the type of stability (weak or strong) nor the fairness objective is part of the preferences themselves. For this reason, the 100 instances generated for each size n are used for both types of stability and for all satisfiability and optimisation versions.

We did not consider instances where all preference lists in all three sets are exact master lists, because the complete set of stable matchings for these instances is known [16]. ML_oneset instances always have a strongly stable matching (Lemma 1), but, contrary to the case with master lists in all three sets, not all solutions have been characterised. Therefore there is value in modelling fairness versions of the problem for instances with this structure.

4.2 Model comparison

In this section, we first test each one of our five models (i.e. HS, DIV-agents, DIV-ranks, UNI-agents, and UNI-ranks) using different heuristic search strategies on small instances in Gecode to find out which strategy performs best. In Gecode experiments, we did not use extra propagation techniques such as lazy clause generation or restarts. Considering that the HS model is implemented in Gecode, and not all search strategies are common to both Gecode and MiniZinc, for a fair comparison between all five models we used indomain_min (assigning the smallest value in the domain) and indomain_max (assigning the largest value in the domain) strategies combined with a search on variables in the given order. The former is referred as *nonemin*, and the latter as *nonemax*. Additionally, we further tested the DIV and UNI models using alternative built-in search strategies that exist in MiniZinc, notably indomain_split, a heuristic that bisects a variable's domain then tries the lower half before trying the upper one. We observed that a strategy that is based on choosing the variable with the smallest domain size using indomain_min results in the best performance for DIV model, while using indomain_split instead of indomain_min leads to the best performance for UNI model. We refer to these as *failmin* and *failsplit* strategies, respectively. Therefore, all the remaining results and plots were obtained by running HS with nonemax strategy, DIV-agents and DIV-ranks with failmin, and UNI-agents and UNI-ranks with failsplit.

During the experiments, we used a time-limit of 10 minutes for each instance. Considering the huge number of all combinations of different parameters in each model, we adapted a look-ahead approach for our tests, i.e. we started performing tests on all models using small instances $n = 4$. Then, we incremented the n for each model that has the potential to be the best. If a model times out on most of the instances for a given combination of parameters, we do not test it further on these instances.

We use notched boxplots [36] in Figures 1, 2, 3, and 4. Figure 1 presents a comparison of total time required by all five models on instances of size $5 \leq n \leq 11$ under weak stability solved using Gecode. The first insight gained is that the HS model handles the instances of small sizes very well. When we examine performance based on the dataset generation methods, we observe that usually a weakly stable matching for the instances in Random is found faster than other datasets when using Gecode. Additionally we observe that the

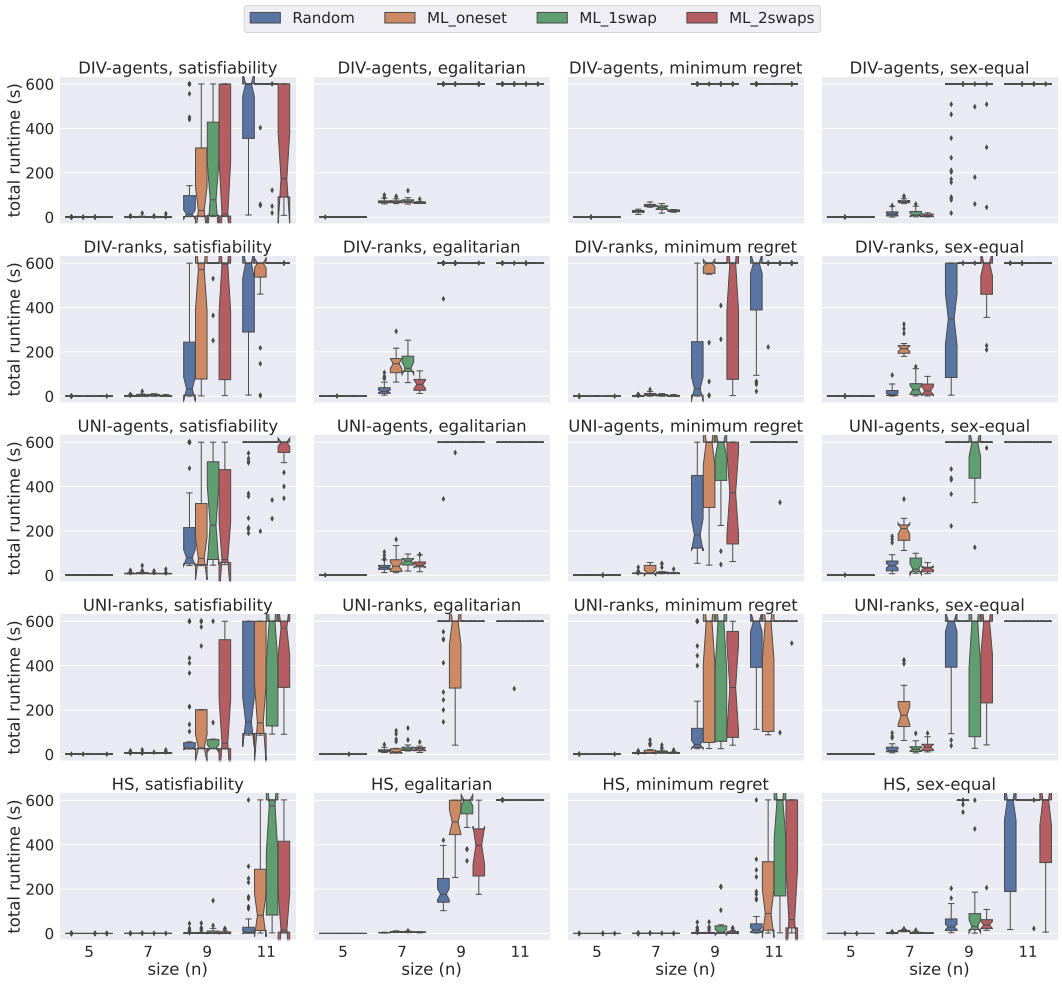


Figure 1 A comparison of total time spent by all models under weak stability using Gecode.

models require more time to solve the instances in ML_1swap for satisfiability, egalitarian, and minimum regret versions. On the other hand, for the sex-equal version of the problem, ML_oneset is the most challenging dataset for all models. We conclude from this figure that HS is the best model when dealing with small instances under weak stability using Gecode.

Figure 2 demonstrates the results obtained from the same datasets under strong stability. Note that the problem model for finding a matching under weak stability is a relaxed version of the model under strong stability. However, it is interesting to observe from the experiments that, on average, strongly stable matchings are found faster than their weak counterpart. We can clearly observe this behaviour on Figure 2. All models except DIV-agents were able to solve all satisfiable instances of size between $4 \leq n \leq 11$ within the given time limit. Therefore, in order to provide more insight into the performance of models, we use a larger scale, i.e. $\{4, 8, 12, 16, 20\}$ in Figure 2. Under strong stability, we clearly observe that HS and DIV-ranks scale better when compared to the other models. For instance, for the satisfiability problem with size $n = 8$, both HS and DIV-ranks quickly solve all the instances. However, both UNI-agents and UNI-ranks require longer time than HS and DIV-ranks, whereas DIV-agents fails to solve many instances within the given time-limit. Both UNI and HS follow the same commitment approach (group commitment). We believe this is hampering their

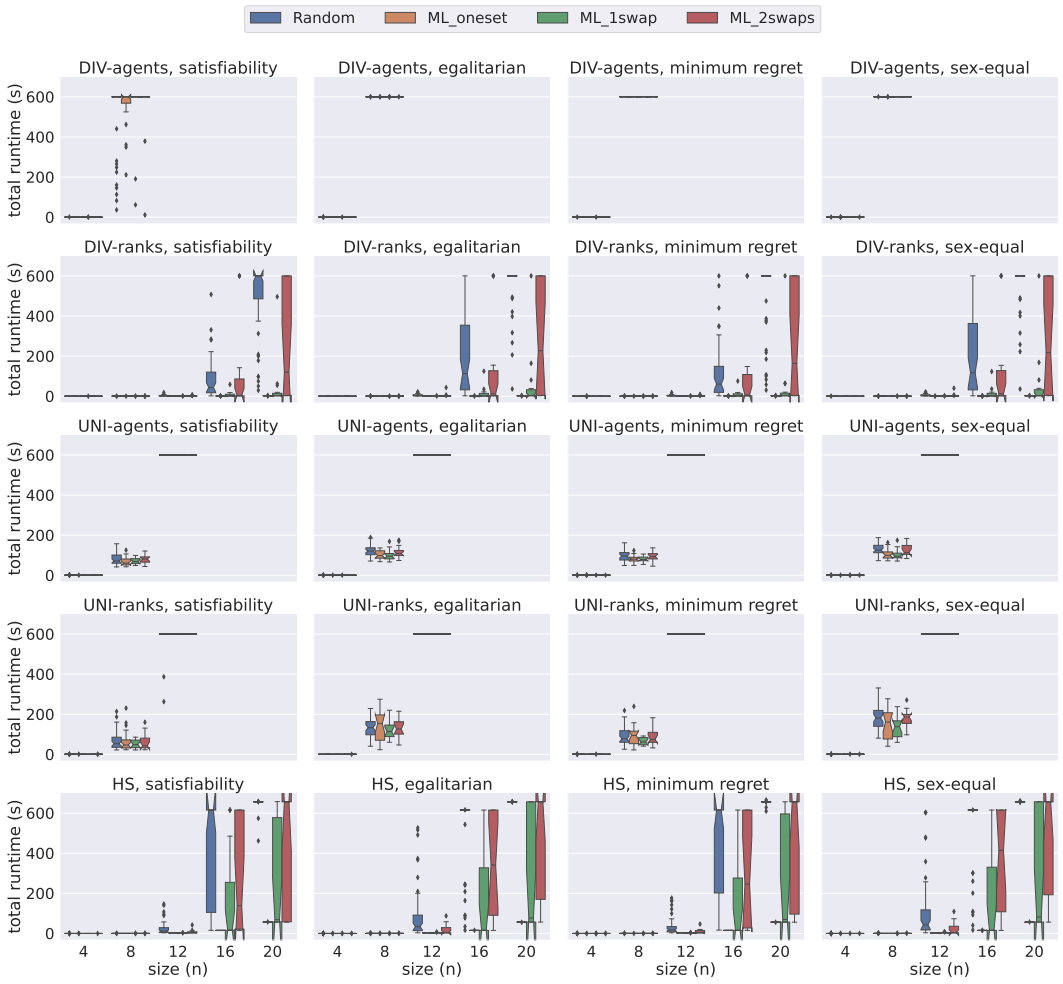


Figure 2 A comparison of total time spent by all models under strong stability using Gecode.

scalability as there are fewer solutions in the strong stability case, which increases the chances of making wrong choices thus leading to higher penalties in the case of group commitment as we have to undo the three pairings. Considering that the time performance of UNI models and DIV-agents are considerably worse for $4 \leq n \leq 12$ when compared to others, we decided to discard them from further experiments. HS is not performing that bad, but its scalability is affected by the computation of the BT sets. The size of each BT set is $O(n^3)$, which is a remarkable overhead when dealing with big instances. We elaborate more on this limitation in the next section. Therefore, we conclude from this figure that DIV-ranks is the most efficient model when working with large n under strong stability using Gecode.

Note that HS was implemented in Gecode directly because it is cheaper to carry out the computation of the BT sets in C++ than in MiniZinc. However, this decision does not put the other models in a disadvantageous position since the main contribution to the running time comes from the solving time and in all cases the solving phase is carried out in C++.

In addition to the Gecode experiments, we tested our four models DIV-agents, DIV-ranks, UNI-agents, and UNI-ranks on Chuffed. Chuffed is the state-of-the art lazy clause solver that performs propagation by recording the reasons of propagation at each step. This helps with efficiently creating nogoods during the search and avoiding failures. Note that due to

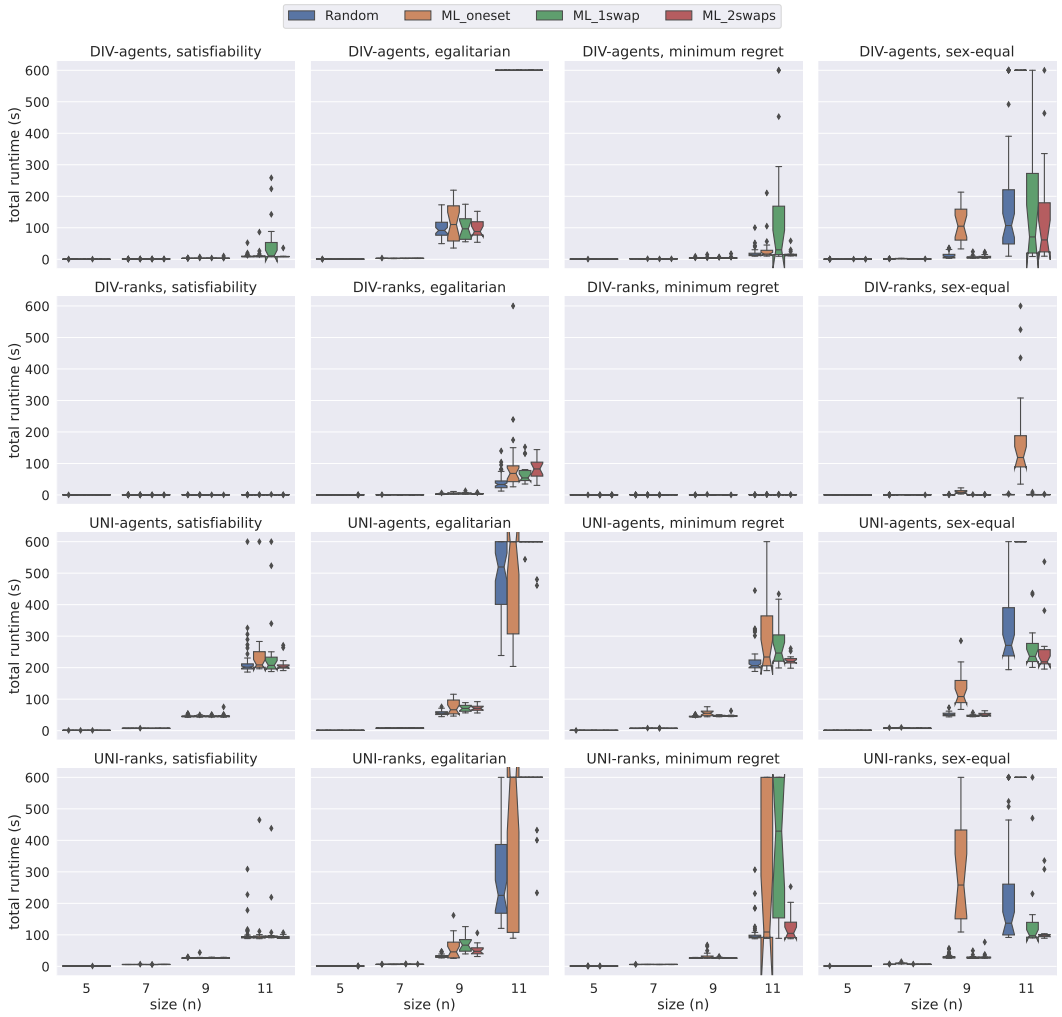


Figure 3 A comparison of total time spent by all models except HS under weak stability using Chuffed.

implementing the HS model directly in Gecode, our Chuffed experiment results do not include the performance of the HS model. Figure 3 presents a comparison of DIV-agents, DIV-ranks, UNI-agents, and UNI-ranks models on instances of size $5 \leq n \leq 11$ under weak stability. In these plots, we can clearly observe that DIV-ranks model has an advantage over the others in terms of total time required to complete the experiments. It is interesting to observe that contrasting with the findings of Gecode experiments in Figure 1, where DIV-agents seems to have an analogous performance with DIV-ranks, if not better, we observe in Figure 3 that DIV-ranks has a clear advantage over DIV-agents when using Chuffed. We believe this shows that DIV-ranks benefits greatly from nogood learning. Additionally, using Figure 3, we can verify our previous observation about ML_oneset being a more challenging dataset generation method for the sex-equal variant under weak stability.

Lastly, Figure 4 demonstrates a comparison of DIV-agents, DIV-ranks, UNI-agents, and UNI-ranks models on instances of size $5 \leq n \leq 11$ under strong stability. A very straightforward intuition of these tests is that the UNI-agents and UNI-ranks models are not able to scale well to larger instances. On the other hand, we observe that DIV-agents handles an increase in

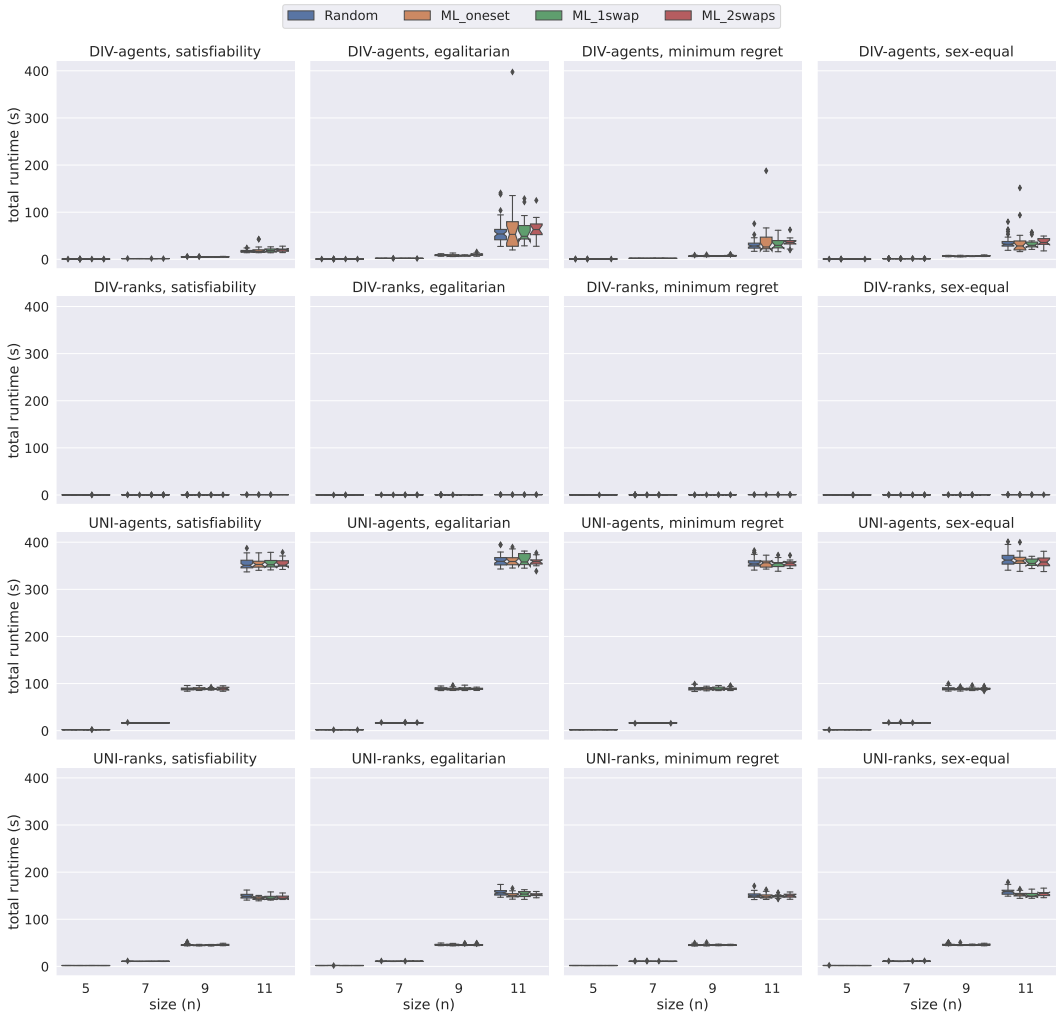


Figure 4 A comparison of total time spent by all models except HS under strong stability using Chuffed.

the number of agents better than the UNI models, but it still performs worse than DIV-ranks when $n \geq 9$. Considering the stable and rapid performance of DIV-ranks using Chuffed and also combining this with our observation on Figure 2, we conclude that the DIV-ranks model is the best one to solve 3DSM-CYC under strong stability.

4.3 Scalability

Considering DIV-ranks is the best performing model in the majority of cases, we performed further experiments using this model on instances with $n \in \{20, 23, 26, 29, 32, 35, 40, 45, 50, 60, 70, 80, 90, 100, 110, 120, 130\}$.

Figure 5 presents a comparison of the median total time required by DIV-ranks using failmin strategy on all four datasets both under weak and strong stability. An interesting insight from this figure is that all four problem variants (i.e. satisfiability, egalitarian, minimum regret, and sex-equal) result in similar performances under strong stability, where instances in Random require the longest time to be solved and ML_oneset requires the least. However,

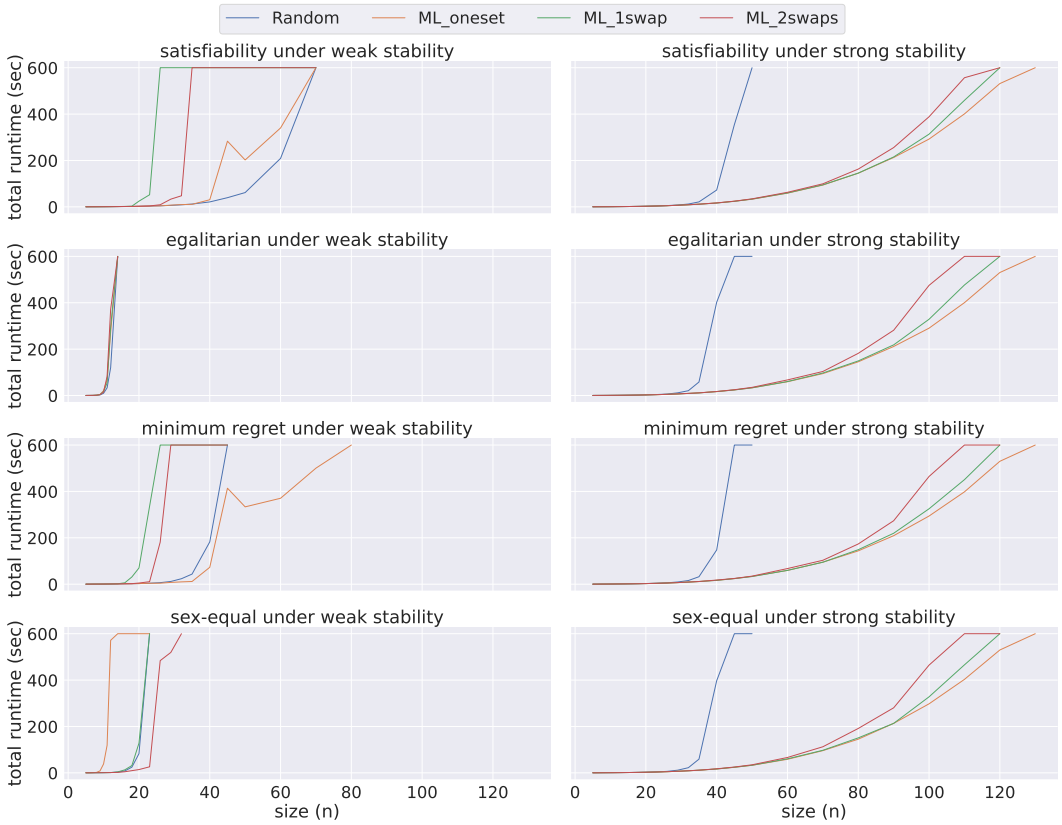


Figure 5 An overview of the performance of DIV-ranks using Chuffed under both weak and strong stability when solving problem instances of different sizes for each dataset.

we cannot make such a generalisation for weakly stable matchings. For instance, ML_oneset dataset is the most challenging dataset for the sex-equal problem variant, but it is also the least challenging for the minimum regret variant under weak stability.

5 Conclusion and future work

We proposed a collection of Constraint Programming models to solve the 3-dimensional stable matching problem with cyclic preferences (3DSM-CYC) using both strong and weak stability notions. Additionally, we extended some well-known fairness notions (egalitarian, minimum regret, and sex-equal) to 3DSM-CYC. The five proposed models are fundamentally different from each other in terms of their commitment (individual or group), and also their domain values (agents or ranks). Our experiments show that nogood learning benefits some models more than others. An unexpected observation is that strong stability turns out to be easier to solve than weak stability. Following a comprehensive empirical evaluation, we conclude that the performances of the proposed models differ with respect to the type of stability and dataset generation method.

The models proposed can be easily adapted to take advantage of the good performance of strong stability by first trying to find a strongly stable matching. Other future work could extend our models to more types of instances, for example by allowing preference lists to be incomplete. One could also look at other redundant constraints in order to best take

advantage of the properties that some instances exhibit with regard to fairness objectives. We remarked that HS pays a high price for the generation of the BT sets but it is possible to generate the sets by demand instead of doing it eagerly.

References

1 D. J. Abraham, A. Levavi, D. F. Manlove, and G. O'Malley. The stable roommates problem with globally-ranked pairs. *Internet Mathematics*, 5:493–515, 2008.

2 Ahmet Alkan. Nonexistence of stable threesome matchings. *Mathematical Social Sciences*, 16(2):207–209, 1988. URL: https://www.sciencedirect.com/science/article/pii/0165489688900534.

3 Michel Balinski and Tayfun Sönmez. A tale of two mechanisms: Student placement. *Journal of Economic Theory*, 84(1):73–94, 1999. URL: https://www.sciencedirect.com/science/article/pii/S0022053198924693.

4 Péter Biró, Robert W Irving, and Ildikó Schlotter. Stable matching with couples: An empirical study. *Journal of Experimental Algorithmics (JEA)*, 16:Article no. 1.2, 2011.

5 Péter Biró and Eric McDermid. Three-sided stable matchings with cyclic preferences. *Algorithmica*, 58(1):5–18, 2010. doi:10.1007/s00453-009-9315-2.

6 Endre Boros, Vladimir Gurvich, Steven Jaslar, and Daniel Krasner. Stable matchings in three-sided systems with cyclic preferences. *Discret. Math.*, 289(1-3):1–10, 2004. doi:10.1016/j.disc.2004.08.012.

7 Sebastian Braun, Nadja Dwenger, and Dorothea Kübler. Telling the truth may not pay off: An empirical study of centralized university admissions in Germany. *The B.E. Journal of Economic Analysis and Policy*, 10, article 22, 2010.

8 Yan Chen and Tayfun Sönmez. Improving efficiency of on-campus housing: an experimental study. *American Economic Review*, 92:1669–1686, 2002.

9 Geoffrey Chu. *Improving combinatorial optimization*. PhD thesis, University of Melbourne, Australia, 2011. URL: http://hdl.handle.net/11343/36679.

10 Sofie De Clercq, Steven Schockaert, Martine De Cock, and Ann Nowé. Solving stable matching problems using answer set programming. *Theory Pract. Log. Program.*, 16(3):247–268, 2016. doi:10.1017/S147106841600003X.

11 Lin Cui and Weijia Jia. Cyclic stable matching for three-sided networking services. *Comput. Networks*, 57(1):351–363, 2013. doi:10.1016/j.comnet.2012.09.021.

12 Vladimir I. Danilov. Existence of stable matchings in some three-sided systems. *Math. Soc. Sci.*, 46(2):145–148, 2003. doi:10.1016/S0165-4896(03)00073-8.

13 Joanna Drummond, Andrew Perrault, and Fahiem Bacchus. SAT is an effective and complete method for solving stable matching problems with couples. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 518–525. AAAI Press, 2015. URL: http://ijcai.org/Abstract/15/079.

14 Pavlos Eirinakis, Dimitris Magos, Ioannis Mourtos, and Panayiotis Miliotis. Finding all stable pairs and solutions to the many-to-many stable matching problem. *INFORMS J. Comput.*, 24(2):245–259, 2012. doi:10.1287/ijoc.1110.0449.

15 Kimmo Eriksson, Jonas Sjöstrand, and Pontus Strimling. Three-dimensional stable matching with cyclic preferences. *Math. Soc. Sci.*, 52(1):77–87, 2006. doi:10.1016/j.mathsocsci.2006.03.005.

16 Guillaume Escamocher and Barry O'Sullivan. Three-dimensional matching instances are rich in stable matchings. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26-29, 2018, Proceedings*, volume 10848, pages 182–197. Springer, 2018. doi:10.1007/978-3-319-93031-2_13.

17 Tomás Feder. A new fixed point approach for stable networks and stable marriages. *J. Comput. Syst. Sci.*, 45(2):233–284, 1992. doi:10.1016/0022-0000(92)90048-N.

18 Tomás Feder. Network flow and 2-satisfiability. *Algorithmica*, 11(3):291–319, 1994. doi:10.1007/BF01240738.

19 Anh-Tuan Gai, Dmitry Lebedev, Fabien Mathieu, Fabien de Montgolfier, Julien Reynier, and Laurent Viennot. Acyclic preference systems in P2P networks. In *Euro-Par 2007, Parallel Processing, 13th International Euro-Par Conference, Rennes, France, August 28-31, 2007, Proceedings*, volume 4641, pages 825–834. Springer, 2007. doi:10.1007/978-3-540-74466-5_88.

20 D. Gale and L. S. Shapley. College admissions and the stability of marriage. *Am. Math. Mon.*, 120(5):386–391, 2013. doi:10.4169/amer.math.monthly.120.05.386.

21 Gecode Team. Gecode: Generic constraint development environment, 2019. Available from http://www.gecode.org.

22 Ian P. Gent, Robert W. Irving, David F. Manlove, Patrick Prosser, and Barbara M. Smith. A constraint programming approach to the stable marriage problem. In *Principles and Practice of Constraint Programming - CP 2001, 7th International Conference, CP 2001, Paphos, Cyprus, November 26 - December 1, 2001, Proceedings*, volume 2239, pages 225–239. Springer, 2001. doi:10.1007/3-540-45578-7_16.

23 Dan Gusfield. Three fast algorithms for four problems in stable marriage. *SIAM J. Comput.*, 16(1):111–128, 1987. doi:10.1137/0216010.

24 Dan Gusfield and Robert W. Irving. *The Stable marriage problem - structure and algorithms*. MIT Press, 1989.

25 Magnús M. Halldórsson, Robert W. Irving, Kazuo Iwama, David F. Manlove, Shuichi Miyazaki, Yasufumi Morita, and Sandy Scott. Approximability results for stable marriage problems with ties. *Theor. Comput. Sci.*, 306(1-3):431–447, 2003. doi:10.1016/S0304-3975(03)00321-9.

26 Chien-Chung Huang. Two's company, three's a crowd: Stable family and threesome roommates problems. In *Algorithms - ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, volume 4698, pages 558–569. Springer, 2007. doi:10.1007/978-3-540-75520-3_50.

27 Robert W. Irving. An efficient algorithm for the "stable roommates" problem. *J. Algorithms*, 6(4):577–595, 1985. doi:10.1016/0196-6774(85)90033-1.

28 Robert W. Irving, Paul Leather, and Dan Gusfield. An efficient algorithm for the "optimal" stable marriage. *J. ACM*, 34(3):532–543, 1987. doi:10.1145/28869.28871.

29 Kazuo Iwama and Shuichi Miyazaki. A survey of the stable marriage problem and its variants. In *International Conference on Informatics Education and Research for Knowledge-Circulating Society (ICKS 2008)*, pages 131–136, 2008.

30 Akiko Kato. Complexity of the sex-equal stable marriage problem. *Japan Journal of Industrial and Applied Mathematics*, 10:1–19, 1993.

31 Donald E. Knuth. *Mariages Stables*. Les Presses de L'Université de Montréal, 1976. English translation in *Stable Marriage and its Relation to Other Combinatorial Problems*, volume 10 of CRM Proceedings and Lecture Notes, American Mathematical Society, 1997.

32 Chi-Kit Lam and C. Gregory Plaxton. On the existence of three-dimensional stable matchings with cyclic preferences. In *Algorithmic Game Theory - 12th International Symposium, SAGT 2019, Athens, Greece, September 30 - October 3, 2019, Proceedings*, volume 11801, pages 329–342. Springer, 2019. doi:10.1007/978-3-030-30473-7_22.

33 David F. Manlove. *Algorithmics of Matching Under Preferences*, volume 2. WorldScientific, 2013. doi:10.1142/8591.

34 David F. Manlove, Gregg O'Malley, Patrick Prosser, and Chris Unsworth. A constraint programming approach to the hospitals / residents problem. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 4th International Conference, CPAIOR 2007, Brussels, Belgium, May 23-26, 2007, Proceedings*, volume 4510, pages 155–170. Springer, 2007. doi:10.1007/978-3-540-72397-4_12.

35 Eric McDermid and Robert W Irving. Sex-equal stable matchings: Complexity and exact algorithms. *Algorithmica*, 68(3):545–570, 2014.

36 Robert Mcgill, John W. Tukey, and Wayne A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978. doi:10.1080/00031305.1978.10479236.

37 Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. Minizinc: Towards a standard CP modelling language. In *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741, pages 529–543. Springer, 2007. doi:10.1007/978-3-540-74970-7_38.

38 Cheng Ng and Daniel S. Hirschberg. Three-dimensional stable matching problems. *SIAM J. Discrete Math.*, 4(2):245–252, 1991. doi:10.1137/0404023.

39 Gregg O'Malley. *Algorithmic aspects of stable matching problems.* PhD thesis, University of Glasgow, UK, 2007. URL: http://theses.gla.ac.uk/64/.

40 Nikita Panchal and Seema Sharma. An efficient algorithm for three dimensional cyclic stable matching. *International Journal of Engineering Research and Technology*, 3(4), 2014.

41 Kanstantsin Pashkovich and Laurent Poirrier. Three-dimensional stable matching with cyclic preferences. *Optim. Lett.*, 14(8):2615–2623, 2020. doi:10.1007/s11590-020-01557-4.

42 Nitsan Perach, Julia Polak, and Uriel G. Rothblum. A stable matching model with an entrance criterion applied to the assignment of students to dormitories at the Technion. *Int. J. Game Theory*, 36(3-4):519–535, 2008. doi:10.1007/s00182-007-0083-4.

43 Neetu Raveendran, Yiyong Zha, Yunfei Zhang, Xin Liu, and Zhu Han. Virtual core network resource allocation in 5G systems using three-sided matching. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

44 Alvin E. Roth and Marilda A. Oliveira Sotomayor. *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis.* Cambridge University Press, 1990. doi:10.1017/CCOL052139015X.

45 Mohamed Siala and Barry O'Sullivan. Revisiting two-sided stability constraints. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 342–357. Springer, 2016.

46 Mohamed Siala and Barry O'Sullivan. Rotation-based formulation for stable matching. In *International Conference on Principles and Practice of Constraint Programming*, pages 262–277. Springer, 2017.

47 Ashok Subramanian. A new approach to stable matching problems. *SIAM J. Comput.*, 23(4):671–700, 1994. doi:10.1137/S0097539789169483.

48 Chris Unsworth and Patrick Prosser. A specialised binary constraint for the stable marriage problem. In *Abstraction, Reformulation and Approximation, 6th International Symposium, SARA 2005, Airth Castle, Scotland, UK, July 26-29, 2005, Proceedings*, volume 3607, pages 218–233. Springer, 2005. doi:10.1007/11527862_16.

49 Chris Unsworth and Patrick Prosser. An n-ary constraint for the stable marriage problem. *CoRR*, abs/1308.0183, 2013. arXiv:1308.0183.

50 Gerhard J. Woeginger. Core stability in hedonic coalition formation. In *Proceedings of SOFSEM 2013, 39th International Conference on Current Trends in Theory and Practice of Computer Science*, volume 7741, pages 33–50. Springer, 2013. doi:10.1007/978-3-642-35843-2_4.

A Tractability proof for ML_oneset

▶ **Lemma 1.** *Each 3DSM-CYC instance with complete lists and one master list admits at least one strongly stable matching.*

Proof. Assume that agent set C is equipped with a master list. For each agent $c_k \in C$, let us relabel the preferred agents of c_k from set A in her preference list so that $\text{rank}_{c_k}(a_i) < \text{rank}_{c_k}(a_{i+1})$ for each $1 \leq i \leq n-1$. We run a serial dictatorship algorithm as follows. First

a_1 chooses her first choice agent in B, who then chooses her first choice agent in C. This triple is removed from the instance. Then we iterate the same starting with a_2, who chooses her first choice among agents in B not yet removed, and so on. Let us relabel agents in B and C so that triples (a_i, b_i, c_i) form the output matching of this algorithm.

First observe that an agent a_i can only prefer an agent b_j to her partner in M if $j < i$. Similarly, a_i prefers b_i to all agents with $j > i$. These observations hold for agents b_i and c_i as well. Each weakly blocking triple thus must include a decrease in the variable index where the strict preference occurs, and simultaneously can contain no decrease in the index elsewhere, because this would make the corresponding agent less satisfied as she was in M. ◀