CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Computing relaxations for the three-dimensional stable matching problem with cyclic preferences
Authors	Cseh, Ágnes;Escamocher, Guillaume;Quesada, Luis
Publication date	2022-07-23
Original Citation	Cseh, Á., Escamocher, G. and Quesada, L. (2022) 'Computing Relaxations for the Three-Dimensional Stable Matching Problem with Cyclic Preferences', CP 2022: 28th International Conference on Principles and Practice of Constraint Programming Haifa, Israel, 31 July-5 August. 16:1-16:19. doi: 10.4230/LIPIcs.CP.2022.16
Type of publication	Conference item
Link to publisher's version	https://doi.org/10.4230/LIPIcs.CP.2022.16 - 10.4230/LIPIcs.CP.2022.16https://drops.dagstuhl.de/opus/volltexte/2022/16645
Rights	© Ágnes Cseh, Guillaume Escamocher, and Luis Quesada; licensed under Creative Commons License CC-BY 4.0 - https://creativecommons.org/licenses/by/4.0/
Download date	2026-11-16 10:06:13
Item downloaded from	https://hdl.handle.net/10468/13413

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Computing Relaxations for the Three-Dimensional Stable Matching Problem with Cyclic Preferences

Ágnes Cseh
Institute of Economics, Centre for Economic and Regional Studies, Budapest, Hungary

Guillaume Escamocher
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

Constraint programming has proven to be a successful framework for determining whether a given instance of the three-dimensional stable matching problem with cyclic preferences (3DSM-CYC) admits a solution. If such an instance is satisfiable, constraint models can even compute its optimal solution for several different objective functions. On the other hand, the only existing output for unsatisfiable 3DSM-CYC instances is a simple declaration of impossibility.

In this paper, we explore four ways to adapt constraint models designed for 3DSM-CYC to the maximum relaxation version of the problem, that is, the computation of the smallest part of an instance whose modification leads to satisfiability. We also extend our models to support the presence of costs on elements in the instance, and to return the relaxation with lowest total cost for each of the four types of relaxation. Empirical results reveal that our relaxation models are efficient, as in most cases, they show little overhead compared to the satisfaction version.

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, relaxation, almost stable matching

Digital Object Identifier 10.4230/LIPIcs.CP.2022.16

Supplementary Material *Software (Source Code)*: `https://doi.org/10.5281/zenodo.6798122`

Funding This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant numbers 12/RC/2289-P2 and 16/SP/3804, which are co-funded under the European Regional Development Fund. Cseh was supported by OTKA grant K128611 and the János Bolyai Research Fellowship.

Acknowledgements COST Action CA16228 European Network for Game Theory.

1 Introduction

Defined on three instead of two agent sets, the 3-dimensional stable matching problem [34] is a natural generalisation of the well-known stable marriage problem [24]. Its most studied variant is the *3-dimensional stable matching problem with cyclic preferences* (3DSM-CYC) [42], 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 finally, agents from the third set only have preferences over agents from the first set.

A matching is a set of triples such that each triple contains one agent from each agent set and each agent appears in at most one triple. 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.


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



Constraint programming approaches allow one to identify instances that do *not* admit a weakly or strongly stable matching – these will be in the focus of our investigation. For such an instance, how to construct a matching that is blocked by only a few triples? Alternatively, which matching minimises the number of justifiably disappointed agents who appear in a blocking triple? A somewhat more sophisticated approach is to assume that a central authority is able to compensate blocking triples or even single agents appearing in blocking triples. If such a compensation has been allocated, then the triple or agent withdraws their claim to form a more advantageous coalition. How to find a matching with the lowest compensation needed to eliminate all blocking triples?

In order to facilitate a general framework, we associate a cost with each agent. The goal is then to minimise the total cost of triples or agents who block the matching, or have to be compensated in order to withdraw from blocking.

1.1 Literature review

We first restrict our attention to related work in the 2-dimensional and non-bipartite stable matching settings. We mention two already established relaxations of stability and also elaborate on problem variants with costs. Then we turn to the 3-dimensional setting, review related work on 3DSM-CYC, and finally discuss constraint models.

1.1.1 Relaxing stability

Various stable matching problems need not admit a stable solution. The relaxation of stability by definition necessarily involves the occurrence of blocking pairs. In the literature, two main relaxations have been defined.

The number of blocking pairs is a characteristic property of every matching. A natural goal is to find a matching with the lowest number of blocking pairs; such a matching is called *almost stable*. This approach has a broad literature: almost stable matchings have been investigated in bipartite [33, 29, 6, 27] and non-bipartite stable matching instances [1, 5, 11, 14], but not in the 3-dimensional setting yet.

Agents who appear in blocking pairs in a solution are called blocking agents. Besides minimizing the number of blocking pairs, another intuitive objective is to minimise the number of blocking agents [49]. The complexity of minimizing the number of blocking agents in a non-bipartite stable matching instance is an open problem that was posed in the seminal book of Manlove [38]. Similar, but slightly more complicated instability measures can be found in the paper of Eriksson and Häggström [19].

1.1.2 Costs and preference negotiation in stable matching problems

Arguably the most natural extension of various matching problems is to consider graphs with edge or vertex costs. For bipartite instances with edge costs, finding a minimum-cost stable matching can be done in polynomial time [31, 28, 21, 22]. The same problem for non-bipartite graphs is NP-hard, but 2-approximable under certain monotonicity constraints using LP methods [53, 54].

Vertex costs play a role in stable matching problems if the agents are part of some type of instance manipulation. In their theoretical study, Boehmer et al. [9] allow agents to reshuffle their preference list. College admission is possibly the most widespread application of stability. Surveys report that bribes have been performed in college admission systems in China, Bulgaria, Moldova, and Serbia [30, 37]. However, preference list manipulation,

potentially done by assigning money to the affected agents, does not imply an illegal action. The internal assignment process of humanitarian organisations [52, 3, 48] aims at stability in the first place, but it also routinely features salary premium negotiations for staff members sent to a less desirable location.

1.1.3 3DSM-CYC

Several applications areas have been modeled by extended 3DSM-CYC instances. Cui and Jia [16] 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. Building upon this work, Panchal and Sharma [44] provided a distributed algorithm that finds a stable solution. Raveendran et al. [47] 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.

A recent real application was described by Bloch et al. [8] who analysed the Paris public housing market. In their work, the first agent set consists of various housing institutions such as the Ministry of Housing, the second agent set is the set of households looking for an apartment, and finally, the third agent set contains the social housing apartments that are to be assigned to these households. Institutions have preferences over household-apartment pairs, and households rank apartments in their order of preference. Cseh and Peters [15] studied a restricted variant where the institutions have preferences directly over the households, no matter which apartment they are matched to.

Maximum relaxations in these applications correspond to the smallest number or cost of users, data sources, servers, households, or housing agencies, who need to be compensated for being part of a blocking triple.

As for the complecity of 3DSM-CYC, Biró and McDermid [7] 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. After a series of papers [10, 20, 45] proving that small 3DSM-CYC instances always admit a weakly stable matching, Lam and Plaxton [36] recently showed NP-hardness for instances with at least 90 agents per agent set – this is also the size of the smallest known no-instance.

1.1.4 CP models for 3DSM-CYC

Several constraint models have been developed for the bipartite stable matching problem and its many-to-one variant [26, 56, 55, 39, 43, 50]. We build upon the recent work of Cseh et al. [13], who introduced five constraint models for 3DSM-CYC. Besides capturing both weak and strong stability, they translated three fairness notions into 3-dimensional matchings.

1.2 Our contribution

In this paper we study four types of relaxation to 3DSM-CYC, based on two established and two new relaxation principles. For each of these types we propose CP approaches that are built on top of the best two approaches from Cseh et al. [13]. We carry out a comprehensive empirical evaluation on a generated data set that includes both satisfiable and unsatisfiable

instances. We analyse the behaviour of our constraint models based on different preference structures, cost functions, and their scalability. The results of the evaluation give insight into the convenience of the introduced types of relaxation, in particular in those cases where the four methods agree on the optimal relaxation.

2 Notation and problem definitions

In Section 2.1 we formally define input and output formats for 3DSM-CYC, using previous notations [13]. The four ways of relaxing stability are then discussed in Section 2.2. Finally, matching costs are introduced in Section 2.3.

2.1 Problem definition

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 is 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 to 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 defines the triple as one already in M.

2.2 Relaxing stability

We examine four different ways to relax stability in 3DSM-CYC. Two of them are standard in the stable matching literature and are based on minimising the number of blocking elements, see Section 2.2.1. The other two relaxation notions are introduced in Section 2.2.2, and they build upon elements that are prohibited to be part of a blocking triple. We remark that all four relaxations can be translated to other stable matching problems as well.

2.2.1 Almost stable matchings

Let $\text{sbt}(M)$ denote the set of strongly blocking triples, and $\text{wbt}(M)$ denote the set of weakly blocking triples to a matching M. Since strongly blocking triples are also weakly blocking, $\text{sbt}(M) \subseteq \text{wbt}(M)$.

▶ **Definition 1.** *A* strong triple-almost stable *(TAS) matching is a matching that minimises the function* $|\text{wbt}(M)|$ *over all matchings* M*. Analogously, a* weak TAS *matching is a matching that minimises the function* $|\text{sbt}(M)|$ *over all matchings* M*.*

If the instance admits a strongly stable matching, then it minimises both functions, but otherwise, there is no connection between the sets of weak TAS and strong TAS matchings.

The agents involved in a strongly blocking triple are called strongly blocking agents, and form the set $\text{sba}(M)$. Analogously, agents involved in any weakly blocking triple are called weakly blocking agents, and form the set $\text{wba}(M)$. Notice that $\text{sba}(M) \subseteq \text{wba}(M)$. A natural objective is to find a matching that minimises the functions $\text{sba}(M)$ or $\text{wba}(M)$.

▶ **Definition 2.** *A matching that minimises* $\text{sba}(M)$ *is called* weak agent-almost stable (AAS)*, while a matching that minimises* $\text{wba}(M)$ *is called* strong AAS*.*

Notice that weak AAS and strong AAS matchings are not identical to weak TAS and strong TAS matchings. As an example, consider two matchings M_1 and M_2 such that $\text{wbt}(M_1) = \{(a_1, b_1, c_1), (a_1, b_1, c_2), (a_1, b_1, c_3)\}$ and $\text{wbt}(M_2) = \{(a_1, b_1, c_1), (a_2, b_2, c_2)\}$. We have $|\text{wbt}(M_1)| = 3$ and $|\text{wbt}(M_2)| = 2$, so M_2 is a better strong TAS candidate than M_1. However $|\text{wba}(M_1)| = |\{a_1, b_1, c_1, c_2, c_3\}| = 5$ and $|\text{wba}(M_2)| = |\{a_1, a_2, b_1, b_2, c_1, c_2\}| = 6$, so M_1 is a better strong AAS candidate than M_2.

2.2.2 Accommodating elements

Instead of minimising the number of blocking elements, we can eliminate them altogether by setting some agents to be *accommodating*. Accommodating agents never report that they are part of a blocking triple, which eliminates all blocking triples containing at least one of those agents. In a realistic scenario, accommodating agents are allocated compensation for their poor match.

▶ **Definition 3.** *A* weak minimally-accommodating stable (MAS) *matching is a matching that minimises the number of accommodating agents needed to eliminate all of its strongly blocking triples. Analogously, a* strong MAS *matching is a matching that minimises the number of accommodating agents needed to eliminate all of its weakly blocking triples.*

Notice that MAS matchings are distinct from AAS matchings. As an example, consider the matchings M_2 from before, where $\text{wbt}(M_2) = \{(a_1, b_1, c_1), (a_2, b_2, c_2)\}$, and the matching M_3 such that $\text{wbt}(M_3) = \{(a_1, b_1, c_1), (a_1, b_2, c_2), (a_1, b_3, c_3)\}$. We have $|\text{wba}(M_2)| = 6$ and $|\text{wba}(M_3)| = 7$, so M_2 is a better strong AAS candidate than M_3. However, we need both an

Table 1 Different ways of interpreting relaxation.

	single agent	more than one agent
minimise the number of blocking elements	agent-almost stable (AAS)	triple-almost stable (TAS)
minimise the number of accommodating elements	minimally-accommodating stable (MAS)	minimally-pair-accommodating stable (MPAS)

agent from $\{a_1, b_1, c_1\}$ and an agent from $\{a_2, b_2, c_2\}$ to be accommodating to eliminate the blocking triples in $\text{wbt}(M_2)$, while setting a single agent, a_1, to be accommodating eliminates all blocking triples in $\text{wbt}(M_3)$. Therefore M_3 is a better strong MAS candidate than M_2.

We can extend the definition of accommodating to groups of agents. Agents x and y from different agent sets form an *accommodating pair* if they are prevented from appearing *together* in a blocking triple. In 3DSM-CYC, exactly one of the two agents has preferences over the other agent, without loss of generality let us assume that it is x. Setting x and y to be an accommodating pair expresses that x receives compensation for not being matched to y specifically. However, x can appear in a blocking triple with another agent from the set of y, and y also can block with any other agent than x. This compensation is thus less powerful than the previous one.

▶ **Definition 4.** *A* weak minimally-pair-accommodating stable (MPAS) *matching is a matching that minimises the number of accommodating pairs needed to eliminate all of its strongly blocking triples. Analogously, a* strong MPAS *matching is a matching that minimises the number of accommodating pairs needed to eliminate all of its weakly blocking triples.*

The sets of MPAS and MAS matchings are incomparable. As an example, consider the matching M_3 from before, where $\text{wbt}(M_3) = \{(a_1, b_1, c_1), (a_1, b_2, c_2), (a_1, b_3, c_3)\}$, and the matching M_4 such that $\text{wbt}(M_4) = (a_1, b_2, c_3), (a_1, b_2, c_2), (a_2, b_3, c_1)$. Only the agent a_1 needs to be accommodating to eliminate all blocking triples in $\text{wbt}(M_3)$, but no single agent appears in all blocking triples of $\text{wbt}(M_4)$, so M_3 is a better strong MAS candidate than M_4. On the other hand, we can eliminate all blocking triples in $\text{wbt}(M_4)$ by setting only two pairs to be accommodating, while we need three to do the same for $\text{wbt}(M_3)$. Therefore M_4 is a better strong MPAS candidate than M_3.

Further extending MPAS to groups of three agents would mean minimising the number of accommodating triples, which is equivalent to TAS.

Table 1 summarises the four different notions of relaxation that we have explored. We remark that while AAS and TAS require that the relaxation set covers every blocking element, for MAS and MPAS, the relaxation set must hit every blocking element.

2.3 Matching costs

When computing a minimal set of elements for relaxation, not all agents might be given equal importance. The central authority might allocate a higher compensation to prioritised blocking pairs or to popular agents. For a given relaxation version, the cost of a matching is the sum of the costs of the elements in the minimal set of this particular relaxation. For a given matching M and arbitrary costs on agents and triples, we thus have for strong stability:

$$\text{Cost}_{\text{AAS}}(M) = \sum_{a \in \text{wba}(M)} \text{Cost}(a)$$

$$\text{Cost}_{\text{TAS}}(M) = \sum_{t \in \text{wbt}(M)} \text{Cost}(t)$$

The definitions for weak stability can be obtained by replacing wbt by sbt and wba by sba. For Cost_{MAS} and $\text{Cost}_{\text{MPAS}}$, we need a further definition.

▶ **Definition 5.** *For a matching M, set S of agents is* agent-convenient *if setting all agents in S to accommodating implies that M is stable. Analogously, set S of pairs of agents is* pair-convenient *for M if setting all pairs in S to accommodating implies the stability of M.*

This definition is the same for both types of stability. We can now write the remaining matching cost definitions for arbitrary agent and pair costs as follows.

$$\text{Cost}_{\text{MAS}}(M) = \min_{S \text{ is agent-convenient for } M} \sum_{a \in S} \text{Cost}(a)$$

$$\text{Cost}_{\text{MPAS}}(M) = \min_{S \text{ is pair-convenient for } M} \sum_{p \in S} \text{Cost}(p)$$

Notice that in all four types of relaxation, not specifying element costs is equivalent to having them all set to 1. We will therefore refer to a relaxation as an *arbitrary-cost relaxation* when elements have an explicit cost, and as a *unit-cost relaxation* when they do not.

3 Methodology

In this section, we explain how we modified the two best performing models for 3DSM-CYC, called DIV-ranks and HS [13], to enable them to deal with soft constraints.

3.1 Soft DIV-ranks model

The DIV-ranks model for 3DSM-CYC with only hard constraints 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 $D(v) = \{1, \ldots, n\}$. Assigning $x_i = j$ (respectively $y_i = j$, or $z_i = j$) corresponds to matching a_i (respectively b_i, or c_i) to her j^{th} preferred agent. The constraints used to find a stable matching M, if any exists, are defined in [13] in the following manner.

- (matching) For all $1 \leq i, j, k \leq n$, the constraint $x_i = \text{rank}_{a_i}(b_j) \wedge y_j = \text{rank}_{b_j}(c_k) \Rightarrow z_k = \text{rank}_{c_k}(a_i)$ is added. This is to ensure that each solution corresponds to a feasible, if not stable, matching. Since domain values correspond to positions in preference lists and not to agents, it is possible for two variables from the same agent set to be assigned the same value. This is why all-different constraints are not used for this model.
- (stability) Under *weak* stability, for all $1 \leq i, j, k \leq n$, the constraint $x_i \leq \text{rank}_{a_i}(b_j) \vee y_j \leq \text{rank}_{b_j}(c_k) \vee z_k \leq \text{rank}_{c_k}(a_i)$ is added. This is to ensure that the triple (a_i, b_j, c_k) is not strongly blocking. 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 $1 \leq i, j, k \leq n$, the constraint $y_j = \text{rank}_{b_j}(c_k) \wedge z_k = \text{rank}_{c_k}(a_i) \Rightarrow x_k = \text{rank}_{a_i}(b_j)$ is added.
- (redundancy) For all $1 \leq i, j, k \leq n$, the constraint $z_k = \text{rank}_{c_k}(a_i) \wedge x_i = \text{rank}_{a_i}(b_j) \Rightarrow y_j = \text{rank}_{b_j}(c_k)$ is added.

For the relaxation version of 3DSM-CYC, we add to the DIV-ranks model an integer variable c_{rel} corresponding to the cost of the relaxation, as well as additional Boolean variables whose exact number depends on the type of relaxation.

- AAS and MAS: a Boolean variable $relA_i$ for each of the n agents a_i in A, a Boolean variable $relB_j$ for each of the n agents b_j in B, and a Boolean variable $relC_k$ for each of the n agents c_k in C, which amounts to $3n$ additional variables.

- TAS: a Boolean variable $rel_{i,j,k}$ for each of the n^3 potential blocking triples (a_i, b_j, c_k).
- MPAS: a Boolean variable $relAB_{i,j}$ for each of the n^2 agent pairs a_i, b_j from $A \times B$, a Boolean variable $relBC_{j,k}$ for each of the n^2 agent pairs b_j, c_k from $B \times C$, and a Boolean variable $relCA_{k,i}$ for each of the n^2 agent pairs c_k, a_i from $C \times A$, which amounts to $3n^2$ additional variables.

For all four types, a variable set to 1 means that its corresponding element is part of the correction set. Determining from the composition of the correction set whether a given triple is allowed to be blocking is expressed in the model by extending the disjunction of the stability constraint corresponding to this triple. The part added depends on the type of the relaxation but not on the kind of stability, so for a given type of relaxation the same part will be added to both weak and strong stability constraints.

- For AAS, we add $\vee(relA_i \wedge relB_j \wedge relC_k)$ to the constraint that checks whether the triple (a_i, b_j, c_k) is blocking. If all three agents are in the correction set, then the constraint is satisfied, and whether this triple is blocking has no effect on the stability of the instance.
- For TAS, we add $\vee rel_{i,j,k}$ to the stability constraint. This immediately satisfies the constraint when the triple is in the correction set.
- For MAS, we add $\vee(relA_i \vee relB_j \vee relC_k)$. Because of the distinction between blocking and accommodating agents, for MAS we only need one agent to be in the correction set for the triple to be disregarded, while for AAS we needed all three agents.
- For MPAS, we add $\vee(relAB_{i,j} \vee relBC_{j,k} \vee relCA_{k,i})$. The constraint is satisfied when any two agents in the triple are present as an accommodating pair in the correction set.

Because relaxation has been added to the stability constraints in a disjunctive way, a trivial solution for the instance can be obtained by assigning 1 to all Boolean variables. Therefore we add a final constraint for the objective function which sums the costs of the elements in the correction set. Minimising this value results in a correction set of minimum cardinality (for unit-cost relaxation), or in a solution of mininum cost (for arbitrary-cost relaxation). Both cases represent a maximum relaxation for the instance. For the unit-cost relaxation, all cost factors in the objective function are replaced by 1.

- For AAS and MAS:
  $c_{rel} = \sum_{i=1}^{n}(relA_i \times \text{Cost}(a_i)) + \sum_{j=1}^{n}(relB_j \times \text{Cost}(b_j)) + \sum_{k=1}^{n}(relC_k \times \text{Cost}(c_k))$.
- For TAS: $c_{rel} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}(rel_{i,j,k} \times (\text{Cost}(a_i, b_j, c_k)))$.
- For MPAS:
  $c_{rel} = \sum_{i=1}^{n}\sum_{j=1}^{n}(relAB_{i,j} \times (\text{Cost}(a_i, b_j))) + \sum_{j=1}^{n}\sum_{k=1}^{n}(relBC_{j,k} \times (\text{Cost}(b_j, c_k)))$
  $+ \sum_{k=1}^{n}\sum_{i=1}^{n}(relCA_{k,i} \times (\text{Cost}(c_k, a_i)))$.

3.2 Soft HS model

We extend the HS model from Cseh et al. [13] by relaxing the constraints that enforce the stability of the matching. Following Cseh et al. [13], in the soft HS model, we assume that T is the set of all possible triples $\{(a_1, b_1, c_1), (a_1, b_1, c_2), \ldots, (a_n, b_n, c_n)\}$, where without loss of generality, the triples in T are ordered, that is, $t_i \in T$ refers to the i^{th} triple of T. We also borrow their definition of non-blocking triples, that is, 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. The variables and constraints of the model are as follows:

- Let M be a set variable whose upper bound is T.
- Let S be a set variable whose upper bound is as follows.
  - For AAS/MAS: $A \cup B \cup C$
  - For TAS: T
  - For MPAS: $A \times B \cup B \times C \cup C \times A$
- Let c be an integer variable corresponding to the cost of the relaxation.
- (matching) Ensure that each agent from each set is matched by having:
  - $\forall a \in A : \sum_{t_i \in T : a \in t_i} (t_i \in M) = 1$;
  - $\forall b \in B : \sum_{t_i \in T : b \in t_i} (t_i \in M) = 1$;
  - $\forall c \in C : \sum_{t_i \in T : c \in t_i} (t_i \in M) = 1$.
- (stability) In the original version, each stable matching is a hitting set of the non-blocking triples (i.e., $\forall t_j \in T : M \cap \{i : t_i \in BT(t_j)\} \neq \emptyset$). We relax this definition as follows.
  - For AAS: $\forall t_j \in T : \exists \langle a, b, c \rangle \in BT(t_j) : \langle a, b, c \rangle \in M \vee \{a, b, c\} \subseteq S$
  - For TAS: $\forall t_j \in T : \exists t_i \in BT(t_j) : t_i \in M \vee t_i \in S$
  - For MAS: $\forall t_j \in T : \exists \langle a, b, c \rangle \in BT(t_j) : \langle a, b, c \rangle \in M \vee \{a, b, c\} \cap S \neq \emptyset$
  - For MPAS: $\forall t_j \in T : \exists \langle a, b, c \rangle \in BT(t_j) : \langle a, b, c \rangle \in M \vee \{\langle a, b \rangle, \langle b, c \rangle, \langle c, a \rangle\} \cap S \neq \emptyset$
- (cost of relaxation) The cost variable is constrained as follows:
  - For AAS/MAS: $c = \sum_{x \in S} Cost(x)$
  - For TAS: $c = \sum_{\langle a,b,c \rangle \in S} Cost(a, b, c)$
  - For MPAS: $c = \sum_{\langle x,y \rangle \in S} Cost(x, y)$

The type of stability is addressed in the computation of the BT sets – the model as such is not concerned with this aspect. In HS, matching M is constrained to be a set of triples representing M as defined in Section 2.1, so the cost of the relaxation follows the definitions in Section 2.3. 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 triple in the mapping. Similarly, S is also represented as an array of Boolean variables. The size of this array is either $3n$, $3n^2$ or n^3, depending on the type of relaxation.

4 Experimental results

All experiments were performed on machines with Intel(R) Xeon(R) CPU with 2.40GHz running on Ubuntu 18.04. Tests for the DIV-ranks model were processed by MiniZinc 2.5.5 [41] before being given to the two constraint solvers Chuffed 0.10.4. [12], which is based on lazy-clause generation, and Gecode 6.3.0 [25]. The HS model on the other hand has been directly encoded using Gecode 6.2.0.

4.1 Dataset

4.1.1 Preference lists

The instances used in our experiments belong to three different classes: Random, ML1swap, and ML2swaps. In the latter two, the preferences are 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 their real-life applications occur in resident matching programs [4], dormitory room assignments [46], cooperative download applications such as BitTorrent [2], and 3-sided networking services [16].

The precise method to create an instance from each class is as follows:

- **Random**: generated randomly from uniform distribution.
- **ML1swap**: all agents in the same agent set follow the same randomly chosen master list. Then in each preference list, the positions of two randomly chosen agents are swapped.

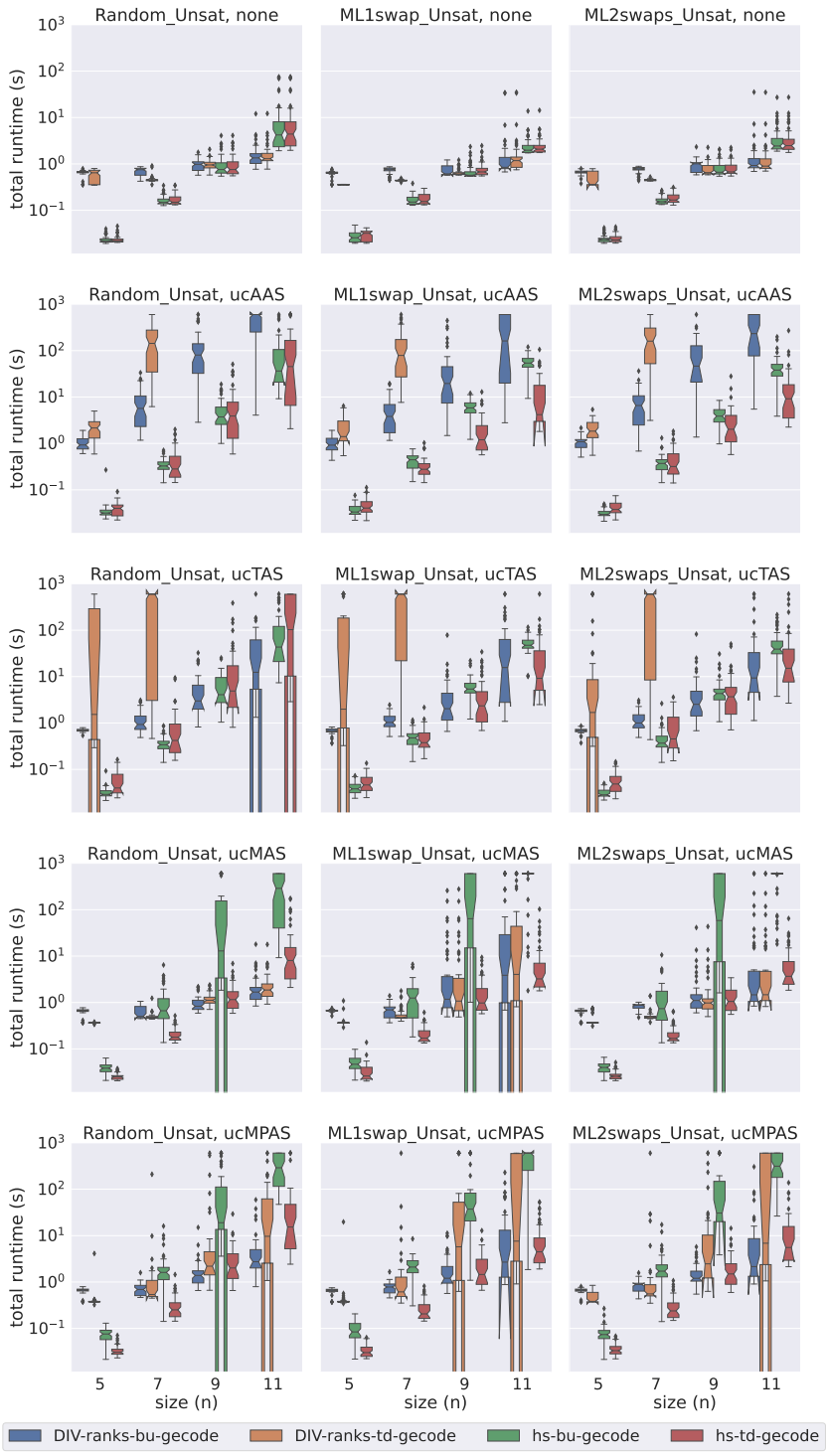


Figure 1 A comparison of total time spent by all Gecode models on the unsatisfiable unit-cost instances.

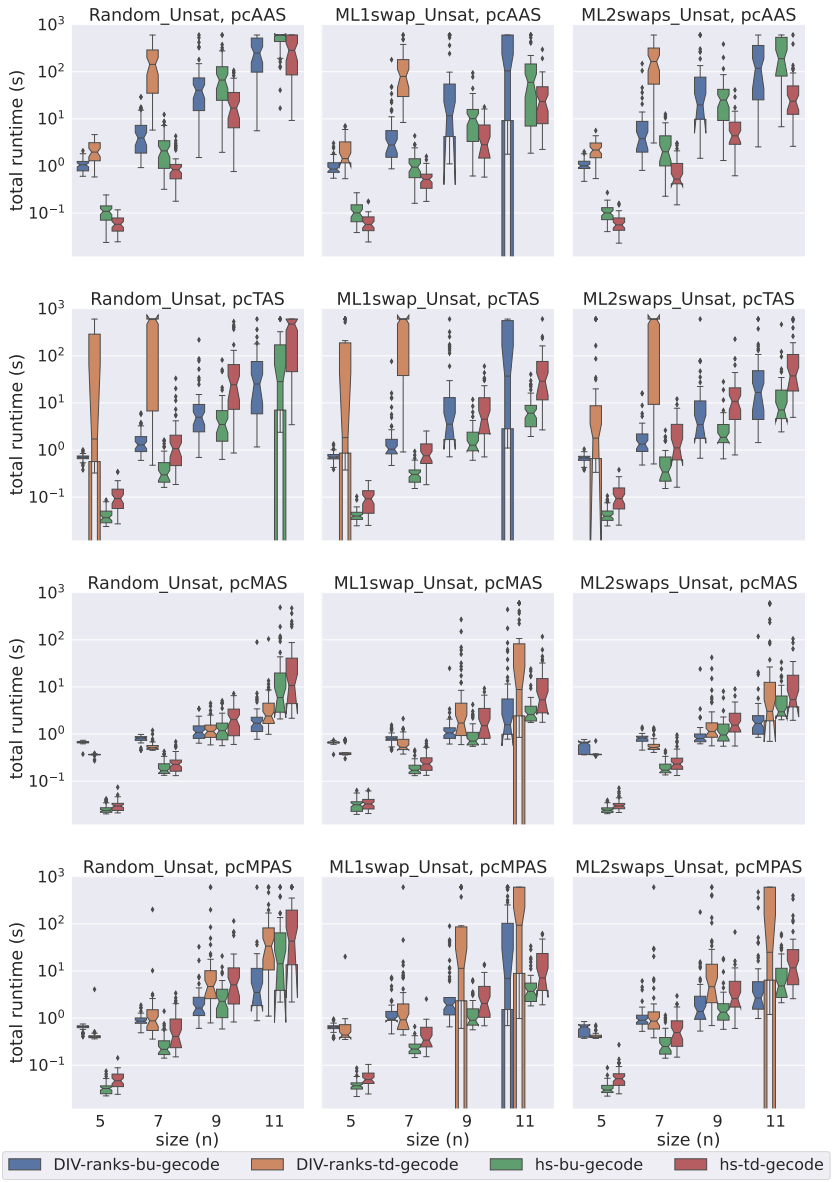


Figure 2 A comparison of total time spent by all Gecode models on the unsatisfiable popularity-cost instances.

- **ML2swaps**: each agent set has a randomly chosen master list that all agents in the set follow. First, two agents are randomly chosen from each agent's preference list, and their positions are swapped. Then, two more agents from each list are randomly chosen such that the new agents were not involved in the first swap, and their positions are swapped.

For each instance class and each odd size $n \in \{5, 7, \ldots, 19\}$, we generated instances with n agents in each agent set, solved the instances under strong stability, and kept the first 50 that were satisfiable and the first 50 that were unsatisfiable. This gave us a total of 300 instances for each size, 150 with a strongly stable matching and 150 without. We had to restrict ourselves to strong stability for unsatisfiability, because the smallest known instance without a weakly stable matching is of size 90 [36], so it would not have been feasible to obtain a representative sample of reasonably-sized unsatisfiable instances for weak stability.

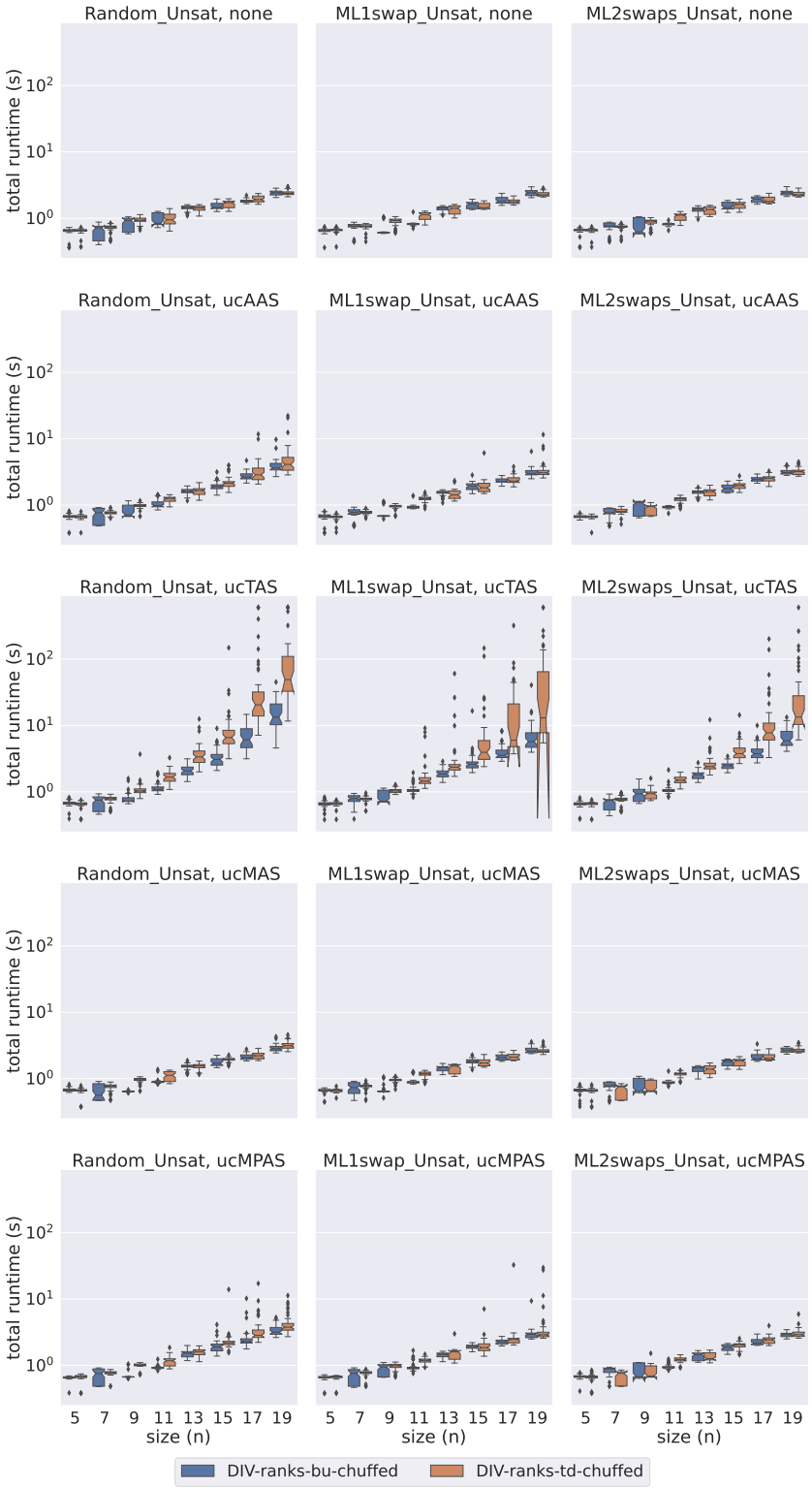


Figure 3 A comparison of total time spent by all Chuffed models on the unsatisfiable unit-cost instances.

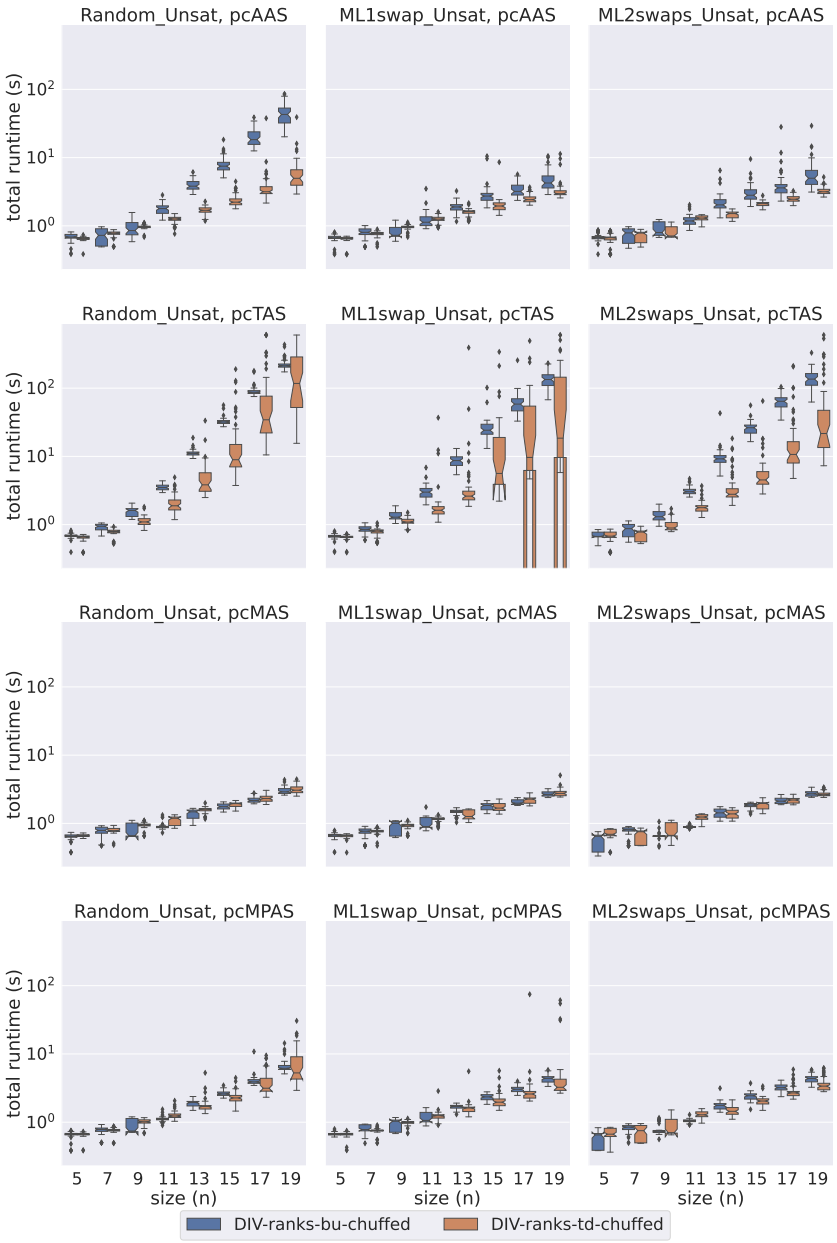


Figure 4 A comparison of total time spent by all Chuffed models on the unsatisfiable popularity-cost instances.

The three types of instances that we studied have been previously used to test the DIV-ranks and HS models, along with a fourth class named ML_oneset [13]. Since ML_oneset instances always admit a strongly stable matching [13], we did not include this additional instance class in our experiments.

4.1.2 Cost formulas

For each configuration of the model, solver, and relaxation type, each instance was set up with two definitions of costs on its elements. The first one is a unit-cost relaxation, corresponding to a cost of 1 for every agent, pair, and triple in the instance. For the second one, that we call *popularity-cost relaxation*, the cost of an agent is a measure of how well she is ranked in other agents' preference lists. Formally the cost of an agent $b \in B$ is defined as:

$$\text{Cost}(b) = \sum_{i=1}^{n} n - \text{rank}_{a_i}(b).$$

The costs of agents from A and C are defined analogously. The intent is to penalise putting popular agents in the correction set, by giving a higher cost to better ranked agents. The cost of a pair (respectively triple) of agents is the sum of the individual costs of the two (respectively three) agents composing it.

4.2 Scalability

In this section we evaluate the performance of DIV-ranks and HS by considering how well they scale with respect to the number of agents in the set. We have decided to classify the experiments into eight groups depending on: (a) the satisfiability of the instance, (b) the solver used and (c) whether the soft constraints have unit cost or not.

The focus of this paper is on dealing with unsatisfiable instances. However, since in practice we cannot always know in advance whether an instance admits a solution, we found it important to check that the satisfiable cases are solved efficiently too. As the instances are satisfiable, the cost of the optimal relaxation is 0 for each one of them, regardless of the type of relaxation. While we could not include the results because of lack of space, all approaches deal with satisfiable instances without major issue.

In Figures 1, 2 3, and 4 we present the results for the unsatisfiable instances. The approaches evaluated are classified in terms of: (a) the model used (DIV-ranks vs HS), (b) the solver used (Gecode vs Chuffed) and (c) the search strategy used (Bottom Up (bu) vs Top Down (td)). *Bottom Up* consists of branching on the cost variable first by selecting the smallest value in the domain first. Effectively this means that we follow a succession of unsatisfiable checks and end with a satisfiable check, which is bound to lead to an optimal solution since we have already proved that there is no solution with a smaller cost. With the *Top Down* strategy we do the opposite: we find a solution and keep on restricting the next one to be better until that is no longer possible. Effectively this means that we follow a succession of satisfiable checks and end with an unsatisfiable check. The unsatisfiable check ensures that the last satisfiable check corresponds to an optimal solution [18].

Our first observation is that the Chuffed approaches clearly outperform the Gecode approaches. As demonstrated by Figures 3 and 4, all Chuffed approaches solve the vast majority of instances of size 15 in less than 10 seconds, while the Gecode approaches struggle with instances of size 11 in quite a few cases. The Chuffed approaches also result in much fewer failures – in some cases the gap is of more than two orders of magnitude.

We consider 9 relaxation types. The first one (none) corresponds to the case where all soft constraints are considered hard. This category was included to gauge the amount of overhead added by modeling each type of relaxation. The other 8 categories correspond to the unit-cost and popularity-cost versions of the four relaxation options introduced in Section 2.2.

In general we observe that our approaches deal much better with MAS and MPAS than with TAS and AAS. In instances where all relaxation types lead to the same optimal relaxation, we can save a considerable amount of time by computing one of our two relaxation types. When it comes to the type of cost, this does not seem to deteriorate much the performance of the Chuffed approaches. In the Gecode approaches we actually observe an improvement in performance when we consider our popularity-cost instances in most of the cases. The situation might be different for instances with completely arbitrary costs.

The Bottom Up vs Top Down comparison is another point where we observe differences between the Chuffed and the Gecode approaches. In the Chuffed approaches, even though in most of the cases we did not observe major differences, in some cases the Top Down exploration led us to visibly better results. The situation in Gecode is quite the opposite. The very same model (DIV-ranks) presented very different behaviours depending on whether Top Down or Bottom Up was used. The Bottom Up tests were completed for all the (small) sizes. However, we had to discard some of the Top Down tests since it was already known that they were going to time out. It is important to remark, though, that the Bottom Up strategy did not always lead to improvements. The improvements were mostly observed when dealing with AAS/TAS instances. Similarly, we observed differences in the performance of HS with respect to the Bottom Up vs Top Down comparison. The Bottom Up strategy led us to better results when dealing with the popularity-cost instances in most of the cases.

We test the scalability of the different relaxation versions on a few large instances in Appendix A.

5 Conclusion and future work

We extended 3DSM-CYC constraint models to four relaxation versions of the problem, two based on already established two-dimensional relaxation notions, and two that we introduced. For each of these four relaxations, we tested our models on instances of various sizes and types, for two different cost functions, and using both a bottom-up and a top-down approach. Our results show that our models are able to efficiently compute a maximum relaxation for unsatisfiable 3DSM-CYC instances.

While our relaxation models performed well for the two cost functions that we studied, it would be interesting to know in what ways their behavior would be affected when given different formulas for the costs of the elements in the instance. For example, one could set the cost of a triple as the difference between the highest and lowest costs of its agents, mirroring the definition of sex-equal [32, 55, 40, 51] optimisation for satisfiable instances. It would be also interesting to find out how the presence of mandatory agents/pairs/triples affect the performance since these constraints are highly motivated in the literature [17, 23, 35].

Another possible avenue of research would be to explore the relations between minimum correction sets of different relaxation types. If for a particular class of instances the maximum relaxations are identical for different types, then one could use our findings that the two new relaxation versions lead to better performance, and search for minimally-accommodating stable matchings instead of almost stable matchings to get the same result faster.

References

1 David J. Abraham, Péter Biró, and David F. Manlove. "Almost stable" matchings in the roommates problem. In Thomas Erlebach and Giuseppe Persiano, editors, *Proceedings of WAOA '05: the 3rd Workshop on Approximation and Online Algorithms*, volume 3879 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

2 David J Abraham, Ariel Levavi, David F Manlove, and Gregg O'Malley. The stable roommates problem with globally-ranked pairs. *Internet Mathematics*, 5:493–515, 2008.

3 Péter Biró. Applications of matching models under preferences. *Trends in Computational Social Choice*, page 345, 2017.

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, David F. Manlove, and Eric J. McDermid. "Almost stable" matchings in the roommates problem with bounded preference lists. *Theoretical Computer Science*, 432:10–20, 2012.

6 Péter Biró, David F. Manlove, and Shubham Mittal. Size versus stability in the marriage problem. *Theoretical Computer Science*, 411:1828–1841, 2010.

7 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.

8 Francis Bloch, David Cantala, and Damián Gibaja. Matching through institutions. *Games Econ. Behav.*, 121:204–231, 2020. doi:10.1016/j.geb.2020.01.010.

9 Niclas Boehmer, Robert Bredereck, Klaus Heeger, and Rolf Niedermeier. Bribery and control in stable marriage. *Journal of Artificial Intelligence Research*, 71:993–1048, 2021.

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

11 Jiehua Chen, Danny Hermelin, Manuel Sorge, and Harel Yedidsion. How hard is it to satisfy (almost) all roommates? In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

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

13 Ágnes Cseh, Guillaume Escamocher, Begüm Genç, and Luis Quesada. A collection of constraint programming models for the three-dimensional stable matching problem with cyclic preferences. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021), Montpellier, France, 25-29 October 2021*, pages 1–19. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021.

14 Ágnes Cseh, Robert W. Irving, and David F. Manlove. The stable roommates problem with short lists. *Theory of Computing Systems*, 63(1):128–149, 2019.

15 Ágnes Cseh and Jannik Peters. Three-dimensional popular matching with cyclic preferences. *CoRR*, abs/2105.09115, 2021. arXiv:2105.09115.

16 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.

17 Vânia M.F. Dias, Guilherme D. Da Fonseca, Celina M.H. De Figueiredo, and Jayme L. Szwarcfiter. The stable marriage problem with restricted pairs. *Theoretical Computer Science*, 306:391–405, 2003.

18 Ulrich Dorndorf, Erwin Pesch, and Toàn Phan-Huy. Solving the open shop scheduling problem. *Journal of Scheduling*, 4(3):157–174, 2001.

19 Kimmo Eriksson and Olle Häggström. Instability of matchings in decentralized markets with various preference structures. *International Journal of Game Theory*, 36(3-4):409–420, 2008.

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

21 Tomás Feder. A new fixed point approach for stable networks and stable marriages. *Journal of Computer and System Sciences*, 45(2):233–284, 1992. doi:10.1016/0022-0000(92)90048-N.

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

23 Tamás Fleiner, Robert W. Irving, and David F. Manlove. Efficient algorithms for generalised stable marriage and roommates problems. *Theoretical Computer Science*, 381:162–176, 2007.

24 David Gale and Lloyd S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 120(5):386–391, 1962. doi:10.4169/amer.math.monthly.120.05.386.

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

26 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.

27 Sushmita Gupta, Pallavi Jain, Sanjukta Roy, Saket Saurabh, and Meirav Zehavi. On the (Parameterized) Complexity of Almost Stable Marriage. In *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020)*, volume 182 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

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

29 Koki Hamada, Kazuo Iwama, and Shuichi Miyazaki. An improved approximation lower bound for finding almost stable maximum matchings. *Information Processing Letters*, 109:1036–1040, 2009.

30 Stephen P. Heyneman, Kathryn H. Anderson, and Nazym Nuraliyeva. The cost of corruption in higher education. *Comparative Education Review*, 52(1):1–25, 2008.

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

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

33 Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127:255–267, 1994.

34 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.

35 Augustine Kwanashie. *Efficient algorithms for optimal matching problems under preferences*. PhD thesis, University of Glasgow, 2015.

36 Chi-Kit Lam and C. Gregory Plaxton. On the existence of three-dimensional stable matchings with cyclic preferences. *Theory of Computing Systems*, pages 1–17, 2021.

37 Qijun Liu and Yaping Peng. Corruption in college admissions examinations in china. *International Journal of Educational Development*, 41:104–111, 2015.

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

39 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.

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

41 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.

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

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

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

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

46 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. *International Journal of Game Theory*, 36(3-4):519–535, 2008. doi:10.1007/s00182-007-0083-4.

47 Neetu Raveendran, Yiyong Zha, Yunfei Zhang, Xin Liu, and Zhu Han. Virtual core network resource allocation in 5g systems using three-sided matching. In *2019 IEEE International Conference on Communications, ICC 2019, Shanghai, China, May 20-24, 2019*, pages 1–6. IEEE, 2019. doi:10.1109/ICC.2019.8762095.

48 Tina Rezvanian. *Integrating Data-Driven Forecasting and Large-Scale Optimization to Improve Humanitarian Response Planning and Preparedness*. PhD thesis, Northeastern University, 2019.

49 Alvin E. Roth and Xiaolin Xing. Turnaround time and bottlenecks in market clearing: Decentralized matching in the market for clinical psychologists. *Journal of Political Economy*, 105(2):284–329, 1997.

50 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.

51 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.

52 Mallory Soldner. *Optimization and measurement in humanitarian operations: Addressing practical needs*. PhD thesis, Georgia Institute of Technology, 2014.

53 Chung-Piaw Teo and Jay Sethuraman. LP based approach to optimal stable matchings. In Michael E. Saks, editor, *Proceedings of SODA '97: the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 710–719. ACM-SIAM, 1997.

54 Chung-Piaw Teo and Jay Sethuraman. The geometry of fractional stable matchings and its applications. *Mathematics of Operations Research*, 23:874–891, 1998.

55 Chris Unsworth and Patrick Prosser. An n-ary constraint for the stable marriage problem. In *Proceedings of the 5th Workshop on Modelling and Solving Problems with Constraints, held at IJCAI '05: the 19th International Joint Conference on Artificial Intelligence*, pages 32–38, 2005.

56 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.

A Scalability on larger instances

To see how well our relaxation models scale on larger instances, we chose the best performing approach for each configuration and ran it on unsatisfiable instances with more than 20 agents in each agent set. These instances, 20 in total, were the ones that were determined

Table 2 Largest solved instance sizes and smallest unsolved instance sizes for each relaxation version when run with a timeout of one hour, using the DIV-ranks model and the Chuffed solver. *: There were two ML1swap instances of size 29 in the dataset. A popularity-cost TAS matching was found before timeout for one but not for the other.

Relaxation	Random		ML1swap		ML2swaps	
	largest solved	smallest unsolved	largest solved	smallest unsolved	largest solved	smallest unsolved
none	35	-	110	-	90	-
unit-cost AAS	35	-	90	110	70	90
popularity-cost AAS	32	35	29	90	70	90
unit-cost TAS	23	29	29	90	35	45
popularity-cost TAS	23	29	29*	29*	29	35
unit-cost MAS	35	-	90	110	70	90
popularity-cost MAS	35	-	90	110	70	90
unit-cost MPAS	35	-	90	110	70	90
popularity-cost MPAS	32	35	90	110	70	90

unsatisfiable for strong stability in the experiments by Cseh et al. [13]. We chose the DIV-ranks model with the Chuffed solver, using the Bottom Up strategy for unit-cost relaxation and Top Down for popularity-cost, because this showed the best performance in our other tests. The results, displayed in Table 2, confirm that it is more efficient to compute MAS relaxations, although AAS and MPAS also scale well for some combinations of instance class and cost function.