Technical Appendix for Polynomial Time Presolve Algorithms for Rotation-Based Models Solving the Robust Stable Matching Problem

Sulian Le Bozec-Chiffoleau, Charles Prud'Homme, Gilles Simonin

Technical Appendix for Polynomial Time Presolve Algorithms for Rotation-Based Models Solving the Robust Stable Matching Problem

Sulian Le Bozec-Chiffoleau[1] , **Charles Prud'homme**[1] and **Gilles Simonin**[1]
[1]TASC, Institut Mines Telecom Atlantique, LS2N UMR 6004, Nantes, France
{sulian.le-bozec-chiffoleau, charles.prudhomme, gilles.simonin}@imt-atlantique.fr

Abstract

This document is the Technical Appendix for the paper "Polynomial Time Presolve Algorithms for Rotation-Based Models Solving the Robust Stable Matching Problem" accepted at IJCAI-2024 [Le Bozec-Chiffoleau *et al.*, 2024]. Section 1 contains the complete proof of a lemma from the IJCAI paper. Section 2 describes the constraints used in the constraint programming model used in the paper, and Section 3 presents the full results of the experiments. In Section 4, there is a link to the GitHub repository containing the Java code for the experiments.

A Complete proof of Lemma 4

Lemma (Increasing property)**.** *Let A, B, C and D be four closed subsets of $\mathcal{R}(\mathcal{I})$. If $A\Delta B \subseteq C\Delta D$ then $d_G(A,B) \leq d_G(C,D)$.*

Proof. Let A, B, C and D be four closed subsets of $\mathcal{R}(\mathcal{I})$ such that $A\Delta B \subseteq C\Delta D$.
A and B are closed subsets so $\forall(\rho_a, \rho_b, \rho) \in (A\Delta B)^2 \times \mathcal{R}(\mathcal{I})$ such that $\rho_a \lhd \rho \lhd \rho_b$, $\rho \in A\Delta B$. We say that $A\Delta B$ is convex. Likewise, $C\Delta D$ is convex. So, if $A\Delta B = C\Delta D$ the proof is trivial. Otherwise, we observe the evolution of the distance value when we add the rotations of $C\Delta D \setminus A\Delta B$ to $A\Delta B$. To do so, we distinguish three sets of rotations (see Fig. 1):

- $\mathcal{V}^- = \{\rho \in C\Delta D \setminus A\Delta B : \exists \rho' \in A\Delta B, \rho \lhd \rho'\}$
- $\mathcal{V}^+ = \{\rho \in C\Delta D \setminus A\Delta B : \exists \rho' \in A\Delta B, \rho' \lhd \rho\}$
- $\mathcal{V}^i = \{\rho \in C\Delta D \setminus A\Delta B : \forall \rho' \in A\Delta B, \rho \in \mathcal{I}(\rho')\}$

By convexity of $A\Delta B$ and $C\Delta D$, those three sets are also convex and $C\Delta D \setminus A\Delta B$ is the disjoint union of the three of them.

As a reminder, each rotation generating a pair is a predecessor of the rotation eliminating it. And each pair has at most one rotation generating it and one rotation eliminating it. Also, each rotation generates as many pairs as it eliminates. Then, each pair eliminated in $\mathcal{V}^-$ can not be generated in $A\Delta B$. Therefore, we do not suppress the pairs that are already present. So:

$$\left|\bigcup_{\rho\in(A\Delta B)\cup\mathcal{V}^-}\mathcal{G}(\rho)\setminus\bigcup_{\rho\in(A\Delta B)\cup\mathcal{V}^-}\mathcal{E}(\rho)\right| \geq \left|\bigcup_{\rho\in A\Delta B}\mathcal{G}(\rho)\setminus\bigcup_{\rho\in A\Delta B}\mathcal{E}(\rho)\right|$$

Also, each pair generated in $\mathcal{V}^+$ can not be eliminated nor generated in $A\Delta B$. Therefore, we suppress at most as much pairs than we add. So:

$$\left|\bigcup_{\rho\in(A\Delta B)\cup\mathcal{V}^+}\mathcal{G}(\rho)\setminus\bigcup_{\rho\in(A\Delta B)\cup\mathcal{V}^+}\mathcal{E}(\rho)\right| \geq \left|\bigcup_{\rho\in A\Delta B}\mathcal{G}(\rho)\setminus\bigcup_{\rho\in A\Delta B}\mathcal{E}(\rho)\right|$$

Given two incomparable meta-rotations, their set of pairs they generate or eliminate are disjoints. So:

$$\left|\bigcup_{\rho\in(A\Delta B)\cup\mathcal{V}^i}\mathcal{G}(\rho)\setminus\bigcup_{\rho\in(A\Delta B)\cup\mathcal{V}^i}\mathcal{E}(\rho)\right| = \left|\bigcup_{\rho\in A\Delta B}\mathcal{G}(\rho)\setminus\bigcup_{\rho\in A\Delta B}\mathcal{E}(\rho)\right| + \left|\bigcup_{\rho\in\mathcal{V}^i}\mathcal{G}(\rho)\setminus\bigcup_{\rho\in\mathcal{V}^i}\mathcal{E}(\rho)\right|$$

We have shown that by adding all the rotations of either $\mathcal{V}^-$, $\mathcal{V}^+$ or $\mathcal{V}^i$ to $A\Delta B$, we can not reduce the value of the distance. Hence, we can prove by a decreasing recurrence on the size of $C\Delta D \setminus A\Delta B$ that $d_G(A,B) \leq d_G(C,D)$. □

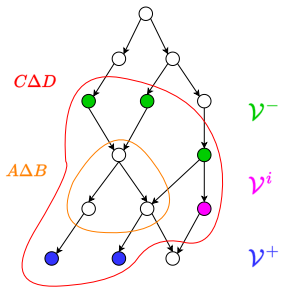

Figure 1: Visual representation of $\mathcal{V}^-$, $\mathcal{V}^+$ and $\mathcal{V}^i$

B Details on two particular constraints

B.1 The constraint CLOSEDSUBSET

Description and properties

The CLOSEDSUBSET constraint is a global constraint over a set variable and is defined as follows:

CLOSEDSUBSET$(\mathcal{S}, H(X)) \equiv$ The set variable $\mathcal{S}$ is a closed subset of the partially ordered space X represented by the Hasse diagram $H(X)$

As a reminder, a closed subset is a subset $S \subseteq X$ for which all predecessors of any element of S are also in S. Below, we state some properties easy to demonstrate that will be needed to filter the constraint:

Lemma B.1.1. *There exists a solution satisfying* CLOSEDSUBSET$(\mathcal{S}, H(X))$ *if and only if* $\forall v \in \underline{\mathcal{S}}, \forall u \in \mathcal{N}^-(v)$, $u \in \overline{\mathcal{S}}$.
Where $\mathcal{N}^-(v)$ denotes the set of predecessors of v resp. to the partial order represented by $H(X)$.

Lemma B.1.2. *An element $v \in \overline{\mathcal{S}}$ does not belong to any solution satisfying* CLOSEDSUBSET$(\mathcal{S}, H(X))$ *if and only if* $\exists u \in \mathcal{N}^-(v)$ *such that* $u \notin \overline{\mathcal{S}}$.

Lemma B.1.3. *An element $u \in \overline{\mathcal{S}}$ belongs to every solution satisfying* CLOSEDSUBSET$(\mathcal{S}, H(X))$ *if and only if* $\exists v \in \mathcal{N}^+(u)$ *such that* $v \in \underline{\mathcal{S}}$.
Where $\mathcal{N}^+(u)$ denotes the set of successors of u resp. to the partial order represented by $H(X)$.

Filtering the constraint
We provide a propagator (Algorithm B.1) for CLOSEDSUBSET and based on Lemmas B.1.2 and B.1.3 that allow to filter the set variable $\mathcal{S}$.

Algorithm B.1 Propagator CLOSEDSUBSET

Require: $\mathcal{S}$, $H(X)$ and $\mathcal{S}^+$ (resp. $\mathcal{S}^-$) the set of elements added (resp. suppressed) to $\mathcal{S}$ since the last call of the propagator
Ensure: Filtering the bounds of $\mathcal{S}$
1: **for** $u \in \mathcal{S}^-$ **do** ▷ Lemma B.1.2
2: **for** $v \in \mathcal{N}^+(u)$ **do**
3: $\overline{\mathcal{S}} \leftarrow \overline{\mathcal{S}} \backslash \{v\}$
4: **for** $v \in \mathcal{S}^+$ **do** ▷ Lemma B.1.3
5: **for** $u \in \mathcal{N}^-(v)$ **do**
6: $\underline{\mathcal{S}} \leftarrow \underline{\mathcal{S}} \cup \{u\}$

Remark. *When the propagator is called for the first time, $\mathcal{S}^- = X \backslash \overline{\mathcal{S}}$ and $\mathcal{S}^+ = \underline{\mathcal{S}}$.*

We denote $N = |X|$. At the creation of the constraint, we compute the predecessors and successors of every element of X, which can be done in $O(N^3)$ time by computing the transitive closure of $H(X)$. Then, the time complexity of Algorithm B.1 is $O(N^2)$.

Correctness and completeness
Algorithm B.1 directly implements the test of the predicates enonciated in Lemmas B.1.2 and B.1.3, which is precised by the comments in the pseudo-code. These lemmas consider all the possible cases of the filtering which are the necessary and sufficient conditions to add an element to the lower bound $\underline{\mathcal{S}}$ and to remove an element from the upper bound $\overline{\mathcal{S}}$. Thus, we can affirm that if we reach a fix point in the filtering of $\mathcal{S}$ relatevely to constraint CLOSEDSUBSET, the filtering is complete.

Furthermore, the two loops in Algorithm B.1 do not interfere with each other, since one decreases $\underline{\mathcal{S}}$ by only observing the elements recently suppressed from $\underline{\mathcal{S}}$, and the other increases $\overline{\mathcal{S}}$ by only observing the elements recently added to $\overline{\mathcal{S}}$. Also, if an element is suppressed from $\underline{\mathcal{S}}$, it means one of his predecessors belongs to $\mathcal{S}^-$ and then every one of its successors will also be suppressed from $\underline{\mathcal{S}}$. Therefore, it is not necessary to call again Algorithm B.1 in order to take into account the recent suppression. A similar reasoning goes for elements added to $\overline{\mathcal{S}}$. Thus, Algorithm B.1 is idempotent and returns a fix point.

In conclusion, Algorithm B.1 does not suppress any potential solution, hence the filtering is correct, and filters as much as possible the set variable $\mathcal{S}$ relatively to constraint CLOSEDSUBSET, hence the filtering is complete.

B.2 The constraint INDEXUNION

Description and properties
The constraint INDEXUNION$(\mathcal{I}, \mathcal{E}, R)$ contains a set variable $\mathcal{I} \subseteq J$ called the set of indices, a set variable $\mathcal{E}$ called the set of elements, and a family of sets $R = (R_i)_{i \in J}$. The constraint is defined as follows:

$$\text{INDEXUNION}(\mathcal{I}, \mathcal{E}, (R_i)_{i \in J}) \equiv \mathcal{E} = \bigcup_{i \in \mathcal{I}} R_i$$

Lemma B.2.1. *There exists a solution satisfying* INDEXUNION$(\mathcal{I}, \mathcal{E}, R)$ *if and only if the following propositions hold:*
- $\forall i \in \underline{\mathcal{I}}, \forall e \in R_i, e \in \overline{\mathcal{E}}$,
- $\forall e \in \underline{\mathcal{E}}, \exists i \in \overline{\mathcal{I}}, e \in R_i$.

Lemma B.2.2. *An element $e \in \overline{\mathcal{E}}$ does not belong to any solution satisfying* INDEXUNION$(\mathcal{I}, \mathcal{E}, R)$ *if and only if* $\forall i \in \overline{\mathcal{I}}$, $e \notin R_i$.

Lemma B.2.3. *An element $e \in \overline{\mathcal{E}}$ belongs to every the solution satisfying* INDEXUNION$(\mathcal{I}, \mathcal{E}, R)$ *if and only if* $\exists i \in \underline{\mathcal{I}}$, $e \in R_i$.

Lemma B.2.4. *An index $i \in \overline{\mathcal{I}}$ does not belong to any solution satisfying* INDEXUNION$(\mathcal{I}, \mathcal{E}, R)$ *if and only if* $\exists e \in R_i$, $e \notin \overline{\mathcal{E}}$.

Lemma B.2.5. *An index $i \in \overline{\mathcal{I}}$ belongs to every the solution satisfying* INDEXUNION$(\mathcal{I}, \mathcal{E}, R)$ *if and only if* $\exists e \in R_i$, $e \in \underline{\mathcal{E}}$ *and* $\forall j \in \overline{\mathcal{I}} \backslash \{i\}$, $e \notin R_j$.

Filtering the constraint
We provide Algorithm B.2 as a propagator of the constraint INDEXUNION and based on Lemmas B.2.2, B.2.3, B.2.4 and B.2.5.

Remark. *When the propagator is called for the first time, $\mathcal{I}^- = J \backslash \overline{\mathcal{I}}$, $\mathcal{I}^+ = \underline{\mathcal{I}}$, $\mathcal{E}^- = (\bigcup_{i \in J} R_i) \backslash \overline{\mathcal{E}}$ and $\mathcal{E}^+ = \underline{\mathcal{E}}$.*

We denote $N = \sum_{i \in J} |R_i|$, the time complexity of Algorithm B.2 is $O(N^2)$.

Correctness and completeness
Algorithm B.2 directly implements the test of the predicates enonciated in Lemmas B.2.2–B.2.5, which is precised by the comments in the pseudo-code. These lemmas consider all the possible cases of the filtering which are the necessary and

Algorithm B.2 Propagator INDEXUNION

Require: $\mathcal{I}$, $\mathcal{E}$, $R = (R_i)_{i \in J}$ and $\mathcal{I}^+$ (resp. $\mathcal{I}^-$) the set of indices added (resp. suppressed) to $\mathcal{I}$ since the last call of the propagator (same for $\mathcal{E}$)
Ensure: Filtering the bounds of $\mathcal{I}$ and $\mathcal{E}$

for $i \in \mathcal{I}^-$ **do** ▷ Lemma B.2.2
  for $e \in R_i$ **do**
    $found \leftarrow FALSE$
    for $j \in \overline{\mathcal{I}}$ **do**
      if $e \in R_j$ **then**
        $found \leftarrow TRUE$
    if $found = FALSE$ **then**
      $\overline{\mathcal{E}} \leftarrow \overline{\mathcal{E}} \backslash \{e\}$
for $i \in \mathcal{I}^+$ **do** ▷ Lemma B.2.3
  for $e \in R_i$ **do**
    $\underline{\mathcal{E}} \leftarrow \underline{\mathcal{E}} \cup \{e\}$
for $e \in \mathcal{E}^-$ **do** ▷ Lemma B.2.4
  for $j \in \overline{\mathcal{I}}$ **do**
    if $e \in R_j$ **then**
      $\overline{\mathcal{I}} \leftarrow \overline{\mathcal{I}} \backslash \{j\}$
for $e \in \mathcal{E}^+$ **do** ▷ Lemma B.2.5
  $found \leftarrow FALSE$
  $index = 0$
  for $j \in \overline{I}$ **do**
    if $e \in R_j$ **then**
      if $found = TRUE$ **then**
        $found \leftarrow FALSE$
        $BREAK$
      else
        $found \leftarrow TRUE$
        $index = j$
  if $found = TRUE$ **then**
    $\underline{\mathcal{I}} \leftarrow \underline{\mathcal{I}} \cup \{index\}$

sufficient conditions to add an element to the lower bounds $\underline{\mathcal{E}}$ and $\underline{\mathcal{I}}$, and to remove an element from the upper bounds $\overline{\mathcal{E}}$ and $\overline{\mathcal{I}}$. Thus, we can affirm that if we reach a fix point in the filtering of $\mathcal{E}$ and $\mathcal{I}$ relatively to constraint INDEXUNION, the filtering is complete.

One call of Algorithm B.2 may not allow to reach a fix point. However, depending on the CP solver we use, it can be done by duplicating the constraint in the CP model, which will allow to call Algorithm B.2 until a fix point is reached.

In conclusion, Algorithm B.2 does not suppress any potential solution, hence the filtering is correct, and allows to filter as much as possible the set variables $\mathcal{E}$ and $\mathcal{I}$ relatively to constraint INDEXUNION when the constraint is duplicated in the model, in this case the filtering is complete.

C Full experimental results

In this section we present the full results obtained during our experiments conducted on SM, HR and MM instances. Unlike the results showed in Section 6, Tables 1–3 contain the statistics on the sizes of $\mathcal{P}_r$, $\mathcal{R}^{up}$ and $\mathcal{R}^{down}$. As for the size of the search space, we notice a great reduction of $\mathcal{P}_r$, $\mathcal{R}^{up}$ and $\mathcal{R}^{down}$ when using Algorithm 1, which also explain the acceleration of LS and CP.

D Source code

The source code used to get the experimental results is presented in the following GitHub repository :

https://github.com/SulianLBC/RSM-IJCAI-2024

A README file details the structure of the project.

References

[Le Bozec-Chiffoleau *et al.*, 2024] Sulian Le Bozec-Chiffoleau, Charles Prud'homme, and Gilles Simonin. Polynomial time presolve algorithms for rotation-based models solving the robust stable matching problem. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 2860–2867. International Joint Conferences on Artificial Intelligence Organization, 8 2024. Main Track.

n	PP	$time$	$\mathcal{P}_r$	$\mathcal{R}^{up}$	$\mathcal{R}^{down}$	$space$	$\bar{b}$	b_{LS}	b_{CP}	$time_{LS}$	$time_{CP}$	$proof_{CP}$
500	1	0.85	1668.4	92.8	92.8	92.8	–	346.2	345.8‡	112.04	4.10	1.29
	1+2	0.95	**6.4**	**2.8**	**2.8**	**5.7**	354.8	346	**345.8**‡	0.15	0.039	0.039
1000	1	6.42	3782.9	150	150	150	–	756.8	755.8‡	191.07	33.27	10.47
	1+2	6.75	**5.7**	**3.8**	**3.1**	**12.1**	778.3	755.8†	755.8‡	0.94	**0.29**	0.02
2000	1	51.99	8831.6	267.3	267.3	267.3	–	1756.6	1637	226.65	380.62	TimeOut
	1+2	53.40	**5.2**	**4.7**	**4.3**	**11.1**	1650	1633	**1632.9**‡	2.62	1.34	0.03

Table 1: Experiments on SM instances (full results)

n-m-c	PP	$time$	$\mathcal{P}_r$	$\mathcal{R}^{up}$	$\mathcal{R}^{down}$	$space$	$\bar{b}$	b_{LS}	b_{CP}	$time_{LS}$	$time_{CP}$	$proof_{CP}$
280-	1	0.08	543	63.2	63.2	63.2	–	127.3	127.2‡	4.64	0.80	0.29
40-7	1+2	0.13	**4.9**	**1.8**	**1.8**	**2.6**	131.3	127.5	**127.2**‡	4.0E-4	0.006	0.005
500-	1	0.22	1293.9	130.2	130.2	130.2	–	289.1†	289.1‡	76.95	5.97	2.451
63-8	1+2	0.33	**5.2**	**3.0**	**2.2**	**6.3**	297.1	289.1†	289.1‡	0.062	**0.037**	0.005
840-	1	0.32	2103.7	195.2	195.2	195.2	–	478.8	477.4‡	84.41	27.50	8.47
70-12	1+2	0.45	**3.6**	**1.7**	**1.6**	**5.2**	485.6	477.7	**477.4**‡	1.5E-3	0.054	0.004

Table 2: Experiments on HR instances (full results)

n-c	PP	$time$	$\mathcal{P}_r$	$\mathcal{R}^{up}$	$\mathcal{R}^{down}$	$space$	$\bar{b}$	b_{LS}	b_{CP}	$time_{LS}$	$time_{CP}$	$proof_{CP}$
100-	1	0.08	677.6	69.3	69.3	69.3	–	154.3†	154.3‡	23.53	1.01	0.39
3	1+2	0.12	**6.8**	**2.2**	**2.8**	**5.8**	162.9	154.3†	154.3‡	**0.005**	0.011	0.010
300-	1	0.53	4478.9	302.6	302.6	302.6	–	986.5	972.1‡	143.08	170.28	49.69
5	1+2	0.96	**5.1**	**3.2**	**3.1**	**9**	984.1	972.2	**972.1**‡	0.63	0.33	0.01
500-	1	2.38	16424.5	879.3	879.3	879.3	–	4183.7	–	109.46	TimeOut	TimeOut
10	1+2	5.43	**4.8**	**2.6**	**2.6**	**14.2**	3504.2	3476.9†	3476.9‡	**0.14**	3.14	0.05

Table 3: Experiments on MM instances (full results)