A Comparative Study of Stability Representations for Solving Many-to-One Matching Problems with Utility-Weighted Objectives, Ties, and Incomplete Lists via Integer Optimization

Pitchaya Wiratchotisatian
Department of Statistics, Khon Kaen University, pitcwi@kku.ac.th

Hoda Atef Yekta
Department of Management, James Madison University, atefyehx@jmu.edu

Andrew C. Trapp
WPI Business School and Data Science Program, Worcester Polytechnic Institute, atrapp@wpi.edu

We consider integer optimization models for finding stable solutions to many-to-one, utility-weighted matching problems with incomplete preference lists and ties. While traditional algorithmic approaches for the stable many-to-one matching problem, such as the Deferred Acceptance algorithm, offer efficient performance for the strict problem setting, adaptation to alternative settings often requires careful customization. Optimization-based approaches are free of the need to create customized algorithms for each unique context, and can readily accommodate such extensions as (incomplete) preference lists with ties; alternative and non-traditional objective functions; and side constraints including those that ensure stable matching outcomes free of waste. We explore the flexibility of optimization-based approaches in several ways. First, we introduce four new constraint sets that prevent justified envy, and a new system of constraints that prevents waste; taken together, they jointly ensure stable matching outcomes. Second, we create two algorithms to accelerate the generation of our proposed constraints. Third, we construct aggregate objective functions to reflect multiple hierarchical emphases by imposing a strict lexicographical order on the individual components. Fourth, we conduct comprehensive experiments to study the computational performance of our proposed optimization models and compare them with models from the extant literature under a variety of problem attributes. Our experiments reveal the circumstances under which each stability representation excels in terms of optimality criteria and computational efficiency on a variety of real and synthetic datasets. One such setting where our proposed stability representations excel includes the important context of when sufficient seats exist for applicants, such as school choice problems and hospital-residency matching.

Key words: Many-to-One Stable Matching; Integer Optimization; Computational Optimization

1. Introduction

This paper studies the many-to-one matching problem in which multiple entities from one set are matched to an entity of another set. The many-to-one matching problem appears in a wide variety of domains such as in school choice (Abdulkadiroğlu and Sönmez 2003,

Abdulkadiroğlu et al. 2006, Kojima and Ünver 2014), college application (Gale and Shapley 1962, Roth and Sotomayor 1989, 1992), hospital-residency matching (Roth 1986, Delorme et al. 2019), and refugee resettlement (Delacrétaz et al. 2019, Ahani et al. 2021). We consider many-to-one matching in the following context: given a set of students $\mathcal{S}$ and a set of project centers $\mathcal{P}$, assign each student s to a project center p, or s is unmatched, considering student preferences over project centers and project center preferences over students under finite project center capacity.

Well-designed matching mechanisms consider several desirable properties such as *efficiency*, *stability*[1] (fairness), and *strategy-proofness*. An *entity* is either a student or a project center director, and a *matching* is a collection of students to project centers such that each student is either assigned to at most one project center or unmatched, and each project center is assigned a set of students of size at most its capacity. Concerning *efficiency*, a matching is *Pareto efficient* if there exists no other matching that *Pareto dominates* it; that is, it is impossible for a reallocation to obtain a better match where at least one entity has improved outcomes without worsening the outcome of other entities. A *blocking pair* is a pair (s,p) in which both student s and project center p prefer one another to their actual assignments or their unassigned (undersubscribed) status. A matching has *justified envy* if there exists a blocking pair (s,p) in which student s prefers project center p to the current assignment and, concurrently, project center p prefers student s over (at least) one of the other students in its current assignment. A matching has *waste* if there exists a blocking pair (s,p) in which student s prefers project center p to the current assignment and, concurrently, project center p has an empty seat and also prefers student s over a vacant position. A matching is *stable* if there exists no blocking pair characterizing either justified envy or waste. A matching mechanism is said to be *strategy-proof* if it is not possible for an entity to obtain a better outcome by misreporting their true preference. While all three properties may be desirable, it has been shown impossible to create a mechanism with all three properties simultaneously (Roth 1982, Alcalde and Barberà 1994). Thus, any matching mechanism that is designed must therefore choose amongst the most desirable properties suiting the particular context.

[1] Stability implies a measure of fairness because in stable matching the occurrence of justified envy is eliminated (Abdulkadiroğlu and Sönmez 2003).

Classical algorithms such as the Deferred Acceptance (DA) (Gale and Shapley 1962) and Top Trading Cycle (TTC) (Shapley and Scarf 1974) algorithms and their variants exist to solve many-to-one matching problems, but extensions to handle ties and incomplete lists efficiently can be challenging. In contrast to the aforementioned algorithmic approaches, our study employs integer optimization (IO) to solve the many-to-one matching problem with incomplete list and ties. IO can readily accommodate side constraints and handle (incomplete) preference lists with ties, and moreover can enable the construction of cohorts with diverse features (Maass et al. 2015, Shimada et al. 2020). Generally speaking, optimization-based approaches are growing increasingly attractive due to their versatile application and competitive computational performance in finding globally optimal solutions via modern computational infrastructure (see, e.g., Bertsimas et al. 2016). Optimization-based approaches can readily produce matching outcomes that maximize efficiency, and can incorporate fairness through the addition of stability constraints (see, e.g., Manlove et al. 2007, Ágoston et al. 2016, Delorme et al. 2019, Shimada et al. 2020), and to date there have been multiple representations of stability in the literature. What has yet to appear in the literature, to the best of our knowledge, is a study that investigates the behavior of multiple representations of stability constraints with respect to their computational performance.

Our paper makes the following contributions. First, we introduce four new constraint sets that each independently forbid justified envy and an additional system of constraints that forbids waste for stable many-to-one matching, and demonstrate that they jointly satisfy the conditions of stability and are equivalent to the state-of-the-art in the literature. Second, we create algorithms to enhance the generation of two proposed stability representations. Third, we develop a lexicographic optimization approach to stable SPC assignment with strictly hierarchical objectives to first maximize total student placements, while secondarily favoring utility. Fourth, we test our mathematical developments by designing and conducting a broad set of computational experiments on our stability representations as well as leading stability representations using real and synthetic datasets, discuss the strengths of various representations across a variety of problem attributes, and highlight the conditions under which various approaches excel.

The remainder of our study is structured as follows. Section 2 covers related background material from the literature. Section 3 introduces our mathematical modeling including

the baseline optimization model of SPC assignment with lexicographic optimization. Section 4 presents the state-of-the-art in stability constraints and then introduces our new representations of stability as well as techniques for their efficient generation. Section 5 presents extensive computational experiments and simulations to compare our approaches with leading stability representations. Section 6 concludes our study.

2. Background

The Deferred Acceptance algorithm (DA) was originally designed for the one-to-one stable marriage problem and, later, extended to the many-to-one college admission problem (Roth 1985). An outcome of the DA algorithm for the stable marriage problem was shown to be stable and efficient for the proposing side (Gale and Shapley 1962). Some properties of the DA algorithm applied to one-to-one matchings do not carry over to many-to-one matchings. Roth (1985) shows that no stable matching mechanism is strategy-proof for the project center side, but student-proposing DA is strategy-proof for students. Another popular approach for many-to-one matching is the Top Trading Cycle algorithm (TTC). Abdulkadiroğlu and Sönmez (2003) explain TTC for school choice and show that TTC is strategy-proof and efficient but does not always yield stable outcomes.

In contrast to standard algorithmic approaches such as DA and TTC, we use techniques from integer optimization (IO) to develop approaches for determining optimal many-to-one matching outcomes. Table 1 summarizes studies that present optimization-based models in the context of stable matching, using various approaches to model the concept of stability.

Vande Vate (1989) is among the first to use optimization to solve the one-to-one stable marriage problem. The study introduces a particular form of constraints that ensure the matching outcomes are stable, and moreover it is shown that they preserve the integrality of the variables representing the assignment of man i to woman j, thus enabling solution via computationally efficient linear programming techniques. Roth et al. (1993) explores the dual and fractional solutions of the stable marriage linear program from Vande Vate (1989). They show that the dual of this particular linear program has a strong relationship to the primal, namely, that each optimal solution to the primal is contained in an optimal solution to the dual, highlighting the duality structure of the underlying linear program.

Baïou and Balinski (2000) are the first to present stability constraints for many-to-one matching. They describe the stable admission polytope — the convex hull or the smallest

Table 1: Optimization-based approaches in the literature that consider stability.

Authors (year)	Type	Optimization Objective	Side Constraints
Vande Vate (1989)	One-to-one	max social values of marriage problem	stability constraints for one-to-one matching describing the stable marriage polytope
Roth et al. (1993)	One-to-one	max utility of marriage problem	stability constraints for one-to-one matching
Baïou and Balinski (2000)	Many-to-one	max utility associated with the assignment	stability constraints for many-to-one matching describing the stable admissions polytope
Kwanashie and Manlove (2014)	Many-to-one	max number of matches	constraint encodings for stable matching
Ágoston et al. (2016)	Many-to-one	min total score-limits or any objective function	score-limits, lower and common quotas, and paired applications
Delorme et al. (2019)	Many-to-one	max number of matches	stability constraint representations in multiple models
Shimada et al. (2020)	Many-to-one	max total utility; min total blocking pairs; min total squared shortages of subgroup members from target (multiobjective)	stability constraints and constraints for matching in groups
Current study	Many-to-one	max number of matches; max student utility; max project center utility	stability constraints (various representations)

convex set of the stable assignments of the college admissions problem. Kwanashie and Manlove (2014) present a set of stability constraints for the Scottish hospital-residency matching with a computational study on their models using both randomly generated and real-world datasets. The form of their presented stability constraints also appears in their recent work in Delorme et al. (2019). Ágoston et al. (2016) focus on the Hungarian admissions problem with four special features: the solution concept of stable score-limits, the presence of lower and common quotas, and paired applications. The score-limit computed for each college is the lowest score that allows students to be admitted. The stable score-limits ensure stable outcomes resulted from using score-limits, in which students are

admitted to their top choice where their score achieved the score limit. Similar to quotas in school choice, lower quotas define the minimum number of assigned students for each college to remain open and upper quotas limit the maximum number of students that can be assigned to each college. The idea of paired applications is similar to many-to-one matching with couples.

Delorme et al. (2019) present integer linear programming (ILP) models for one-to-one pairing of children with adoptive families and many-to-one hospital-residency assignment, each of which maximizes the number of stable matches. They also derive an enhanced model that reduces the number of nonzero elements with respective forms of stability constraints and conduct experiments on real and random datasets. Shimada et al. (2020) also consider many-to-one matching for hospital-residency. They address the trade-off between stability and efficiency through the use of multiple objectives.

In many real-life applications, it is unrealistic to require preferences over every entity on the other side, and moreover it may be the case that an entity has ambivalence in their preferences over multiple entities. Optimization-based approaches excel in such contexts of matching with incomplete lists and ties among preferences. Our study solves the more general problem that allows for weighted matches in the context of incomplete lists and ties. In particular, students can have ties over project centers, while project centers can also have ties among students (though rarely do).

3. Mathematical Modeling and Formulations

Our study is most related to the specific Hospitals/Residents problem with Ties problem (*HRT*) described in Kwanashie and Manlove (2014), which models the many-to-one matching of residents to hospitals with ties and incomplete lists. A key distinction between SPC assignment and *HRT* is that, while both seek to maximize the number of matches, our SPC assignment also considers weighted utilities for one or both sides of the matching. Our problem may be called *MAX-WT-HRT-GRP* where GRP stands for Globally Ranked Pairs in which each matching pair has a weight assigned and the preference lists can be derived from ordering these weights. We call our problem *HRT-W* for brevity. We note that *HRT-W* reduces to HRT when all utility weights in the objective function are zero. We first describe the *HRT-W* problem for matching students to project centers, and subsequently present a multi-objective optimization model that is used as a baseline throughout this study.

3.1. Description and Assumptions of the *HRT-W* Problem

The student-project center market is defined by a finite set $\mathcal{S}$ of students together with a finite set $\mathcal{P}$ of project centers, each with a capacity of c_p seats. Student $s \in \mathcal{S}$ and project center $p \in \mathcal{P}$ have a *weak* preference order $\succeq$ or *strict* preference order $\succ$ over one another. The need for weak preferences arises from indifference between entities, which leads to ties in preference lists. Because some students may be unmatched, we augment the set of project centers $\mathcal{P}$ used in our model formulation to include a *virtual* project center p_0 dedicated to otherwise unassigned students[2] with (unconstrained) capacity $|\mathcal{S}|$. The *ranking level of student* s in the preference list of a project center is defined by a finite set $\mathcal{R}_s$, whereas the *ranking level of project center* p in the preference list of a student is defined by a finite set $\mathcal{R}_p$, both indexed by r. *Eligible* assignments consist of students to project centers that each rank one another on their preference lists. Moreover, we assume that it is preferable for a project center to accept any eligible student over an empty seat.

Preferences can be defined as ordinal or cardinal utility values. The utility values of the *cardinal* preference can be expressed numerically and the difference in magnitude between entities matters. In contrast, the utility values of the *ordinal* preference cannot be expressed numerically and the difference in magnitude between entities is irrelevant. Our study considers cardinal preferences. Let q_{sp} represent the cardinal value of project center p for student s, and for two project centers p_i and p_j, $p_i \succeq_s p_j \iff q_{sp_i} \geq q_{sp_j}$. Each project center p has a cardinal preference value k_{sp} for assigning one of its seats to student s calculated based on a variety of observable student characteristics. We define the value of assigning student s to project center p, also known as the market utility, as $u_{sp} := U(q_{sp}, k_{sp})$, which is a linear function of cardinal preferences of student s and project center p. Define $\mathcal{P}_s$ as the set of project centers that are ranked by student s. Students prefer being unmatched to being assigned to undesirable project centers that are not in their list, that is, $q_{sp} > q_{sp_0} > 0 \ \forall \ s \in \mathcal{S}, \ \forall \ p \in \mathcal{P}_s$. Moreover, a virtual project center prefers all students equally, $k_{sp_0} = 0 \ \forall \ s \in \mathcal{S}$, and assigning students to a virtual project center does not contribute to the objective function, that is, $u_{sp_0} = 0 \ \forall \ s \in \mathcal{S}$. Further explanation can be found in Section 5.2. With the introduction of binary decision variables

[2] It is inevitable that matching markets with incomplete preference lists may have unmatched students. For example, suppose there are three students with only one project center p in their preference list and there are two project centers each with capacity of two. In this scenario, although there are enough seats for all three students, a student who is least preferred by p would be unmatched.

$x_{sp}, \forall\ s \in \mathcal{S},\ p \in \mathcal{P}$ that take value 1 if student s is assigned to project center p, and 0 otherwise, we can present our optimization model. Table 2 contains all model notation.

Table 2: Notation for matching student to project center models.

Sets	
$\mathcal{S}$	Set of students, indexed by s
$\mathcal{P}$	Set of project centers including a virtual project center p_0, indexed by p
$\mathcal{R}_s$	Set of ranking levels of student s over the project center preference list, indexed by $r \leq \lvert\mathcal{S}\rvert$
$\mathcal{R}_p$	Set of ranking levels of project center p over the student preference list, indexed by $r \leq \lvert\mathcal{P}\rvert$
$\mathcal{S}_p$	Set of students for whom project center p ranks in its preference list, indexed by s
$\mathcal{P}_s$	Set of project centers that student s ranks in their preference list, indexed by p
$\mathcal{S}_p^{(r)}$	Set of students for whom project center p ranks in level $r \in \mathcal{R}_s$, indexed by $s^{(r)}$
$\mathcal{P}_s^{(r)}$	Set of project centers that student s ranks in level $r \in \mathcal{R}_p$, indexed by $p^{(r)}$
Parameters	
u_{sp}	Utility value of matching student s to project center p
q_{sp}	Student preference of student s at project center p
k_{sp}	Director preference of student s at project center p
c_p	Capacity of project center p
Decision Variables	
x_{sp}	Binary variable assuming a value of 1 if student s is assigned to project center p; 0 otherwise
z_p	Binary indicator variable with value of 1 if project center p has at least one empty seat; 0 otherwise
$\boldsymbol{x}$	Entire $\lvert\mathcal{S}\rvert \times \lvert\mathcal{P}\rvert$ matching outcome

3.2. Baseline Optimization Formulation

We present a lexicographical multi-objective optimization formulation for the student-project center matching problem, in which both the total number of students placed and the total market utility are prioritized. It is assumed that maximizing the total number of student placements is paramount. Hence, we design a novel objective function that ensures total market utility is considered in a strictly subordinate manner, that is, it is only used to break ties among (stable) solutions for which the number of matches is maximized.

$$\text{maximize} \quad \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} x_{sp} + \gamma \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} u_{sp}x_{sp} \tag{1a}$$

$$\text{subject to} \quad \sum_{p\in\mathcal{P}} x_{sp} = 1 \quad \forall\ s \in \mathcal{S}, \tag{1b}$$

$$\sum_{s\in\mathcal{S}} x_{sp} \leq c_p \quad \forall\ p \in \mathcal{P}, \tag{1c}$$

$$x_{sp} = 0 \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}_s : p_0 \succ_s p, \tag{1d}$$

$$[\ \textit{Blocking pair elimination constraints}\ ], \tag{1e}$$

$$x_{sp} \in \{0,1\} \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}. \tag{1f}$$

Objective function (1a) maximizes the sum of total student placement and weighted total market utility. We carefully select the value of γ in such a way that our objective function strictly prioritizes the maximization of total student placement first, and then utility.[3] Constraint set (1b) requires that each student must be assigned to exactly one project center, whether a project center in the preference list, or the virtual project center p_0 introduced for unmatched students. Constraint set (1c) ensures that the number of students assigned to each project center does not exceed site capacity. Constraint sets (1d) and (1e) together ensure stability; constraint set (1d) preserves individual rationality, while constraint set (1e) prohibits solutions with blocking pairs. In Section 4 we introduce several new many-to-one stability constraint representations and a set of constraints that prevent blocking pairs so as to ensure stable matching outcomes, and discuss their relationship to existing forms in the literature. Variable domains are stated in (1f). We will extend this baseline formulation to create other models through our study.

4. New Representations of Many-to-One Stability

Optimization formulation (1) features a single lexicographic objective function that preemptively maximizes the total number of placed students while secondarily favoring utility and yet maintaining stability. We maintain strict conditions of stability through constraints that express the prevention of blocking pairs. Recall that a match is *stable* if there exists no blocking pair characterizing justified envy or waste. This definition has already been represented in several mathematical forms for both one-to-one (see, e.g., Vande Vate 1989, Roth et al. 1993) and many-to-one matching markets in the literature (see, e.g., Baïou and Balinski 2000, Delorme et al. 2019). To set the stage for the presentation of our new and alternative linear forms of stability that eliminate both justified envy and waste, we first review the existing stability constraints in the literature.

[3] We set coefficient γ equal to the reciprocal of total available capacity at project centers plus a small positive value to ensure that the weighted total market utility is strictly less than one, thereby ensuring any single additional placement is preferred over the greatest market utility.

4.1. Existing Stability Representations

Vande Vate (1989) introduces a binary integer programming formulation in the context of the one-to-one stable marriage problem and appears to be the first to model stability. The study assumes a set of men $\mathcal{M}$ and a set of women $\mathcal{W}$ with $|\mathcal{M}| = |\mathcal{W}|$, and the existence of complete preference lists with strict ordering over all candidates on the other side. Let $x_{mw} = 1$ if man m is matched to woman w; 0 otherwise. We present their associated constraint set in (VV), which imposes stability by requiring that if woman w matches someone less desirable than man m, then m must match someone more desirable than w:

$$\sum_{i \prec_w m} x_{iw} - \sum_{j \succ_m w} x_{mj} \leq 0 \quad \forall\ m \in \mathcal{M},\ \forall\ w \in \mathcal{W}. \tag{VV}$$

$$\sum_{j \prec_s p} x_{sj} - \sum_{i \succ_p s} x_{ip} \leq 0 \quad \forall\ p \in \mathcal{P},\ \forall\ s \in \mathcal{S}. \tag{2}$$

Baïou and Balinski (2000) appear to be the first to use a set of linear constraints to express the concept of stability for many-to-one matching, doing so in the context of complete preference lists and strict ordering of preferences. Their constraint set (BB) imposes stability by enforcing that if student s is not assigned to project center p ($x_{sp} = 0$), then either student s is assigned to project center j that is preferred to p, or all of the c_p seats of project center p are assigned to students that p prefers to s:

$$c_p x_{sp} + c_p \sum_{j \succ_s p} x_{sj} + \sum_{i \succ_p s} x_{ip} \geq c_p \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}. \tag{BB}$$

Adapted to the context with ties, (BB) can be modified to (BBT) whereby stability is imposed by also considering the assignment of student s to other project centers ranked identically to project center p and the assignment of other students ranked identically to student s to project center p, when s is not assigned to p:

$$c_p x_{sp} + c_p \sum_{\substack{j \neq p: \\ j \succeq_s p}} x_{sj} + \sum_{\substack{i \neq s: \\ i \succeq_p s}} x_{ip} \geq c_p \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}. \tag{BBT}$$

Constraint set (BBT) resembles the constraint set presented in Kwanashie and Manlove (2014) as well as Delorme et al. (2019), who present integer optimization frameworks that embed linear stability constraints in the context of many-to-one matching with ties and incomplete lists. Their constraint set presented as (HRT) below imposes the condition of stability by ensuring that if student s is not assigned to project center p or any project

center that s prefers at least as much as p (that is, $\sum_{j \succeq_s p} x_{sj} = 0$), then project center p must fill its capacity with students it ranks at the same level or higher than s:

$$c_p \left(1 - \sum_{j \succeq_s p} x_{sj}\right) \leq \sum_{i \succeq_p s} x_{ip} \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}. \tag{HRT}$$

We note the similarity between (BBT) and (HRT), as (HRT) can be derived from (BBT) when generalizing to ties and incomplete lists, further details of the proof are in Appendix A. As limited computational testing reveals that both (BBT) and (HRT) exhibit similar computational performance, we elect to proceed with (BBT) hereafter.

4.2. Proposed Stability Representations

A stable matching has no justified envy or waste. We introduce constraint sets that eliminate blocking pairs associated with justified envy, along with a system of constraints to eliminate blocking pairs associated with waste. Our new stability representations have two origins. First, we extend the one-to-one stability constraint set of Vande Vate (1989) to obtain a new stability constraint set for the many-to-one context; combining with our system that eliminates blocking pairs associated with waste, (VVM) is sufficient to ensure stable outcomes in the many-to-one context. Second, we introduce a new pairwise form of linear constraints to prevent blocking pairs associated with justified envy. We then derive two arguably more useful and related forms through aggregation, and introduce algorithms to enhance the computational performance of their construction.

Extending (VV) to Eliminate Justified Envy in the Many-to-One Context. We derive a many-to-one stability representation from the one-to-one stability constraint (VV) of Vande Vate (1989). Viewing men (m) as students (s) and women (w) as project centers (p), we extend in (VVM) the concept of stable marriage to the many-to-one context with ties among preferences and incomplete lists where unmatched students are placed in the virtual project center p_0. The derivation of (VVM) appears in Proposition 1 of Appendix A.

$$\sum_{i \prec_p s} x_{ip} \leq c_p \left(1 - \sum_{j \prec_s p} x_{sj}\right) \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}. \tag{VVM}$$

Constraint set (VVM) ensures that if student s is assigned to a project center less desirable than p, then project center p cannot be assigned a student less preferable than s. It also implies that if project center p has its capacity filled by students less desirable than s, then student s cannot be assigned to a center less preferable than p; otherwise, a matching pair (s, p) could block the market and leave their assigned match.

Introducing A System of Constraints to Ensure Outcomes Without Waste. Stability in the many-to-one context requires not only the elimination of blocking pairs associated with justified envy, but also the elimination of blocking pairs associated with waste. A blocking pair (i,p) associated with waste occurs when student i prefers project center p to their match and project center p prefers student i to an empty seat. We introduce a system of constraints that sense whether project center p has empty seats via an indicator variable z_p, using z_p to forbid such blocking pairs for our proposed stability constraint sets.

The number of empty seats at each project center is the difference between the capacity and the number of students assigned to the center. Define z_p as an indicator that takes value of 1 if project center p has at least one empty seat ($c_p - \sum_{s\in\mathcal{S}} x_{sp} > 0$). We introduce the (NWS) system of constraints to prevent blocking pairs associated with waste:

$$z_p \leq c_p - \sum_{s\in\mathcal{S}} x_{sp} \leq c_p z_p \quad \forall\ p\in\mathcal{P}, \tag{NWa}$$

$$\sum_{i\in\mathcal{S}_p}\sum_{\substack{j\in\mathcal{P}:\\ j\prec_i p}} x_{ij} \leq \mathcal{M}_w(1-z_p) \quad \forall\ p\in\mathcal{P}, \tag{NWb}$$

$$z_p \in \{0,1\} \quad \forall\ p\in\mathcal{P}. \tag{NWc}$$

Constraint set (NWa) ensures that $z_p = 1$ if there is at least one empty seat at project center p, that is, $c_p - \sum_{s\in\mathcal{S}} x_{sp} > 0$, and 0 otherwise. Constraint set (NWb) prevents blocking pair (i,p) when waste exists by ensuring that if there exists an empty seat at project center p ($z_p = 1$), then there exists no match $(i,j)\in\mathcal{S}\times\mathcal{P}$ in which project center p prefers student i to an empty seat and student i prefers project center p to project center j. We set the upper bound of the number of blocking pairs at project center p, $\mathcal{M}_w$, to be the number of students who rank p in their preference list. Variable domains are covered in (NWc).

The (NWS) system eliminates blocking pairs associated with waste. They complement (VVM) and all of our subsequently introduced stability constraint sets that eliminate blocking pairs associated with justified envy, thereby ensuring stability. However, in the presence of an objective function that prioritizes student utility, (NWS) becomes superfluous and may be dropped. This special context induces stability because prioritizing student utility incentivizes assigning students to their best available choice. As any desirable project centers prefer accepting students in their list to having empty seats, the outcomes will have no empty seats that students would value over their own assignments and, thus, result in stable matching outcomes.

Introducing New Stability Representations To Eliminate Justified Envy. We now introduce three linear representations of stability that, similar to (VVM), forbid blocking pairs associated with justified envy. We then use aggregation to derive two new linear representations of stability. We subsequently present two algorithms to improve their generation and performance based on the order of aggregation.

The notion of justified envy in stable markets states that pairs (s,p) and (i,j) cannot be simultaneously matched if either of the blocking pairs (s,j) or (i,p) exist in which their members prefer one other to the actual match. In other words, if student s prefers project center j to p and project center j also prefers student s to i, then matching pairs (s,p) and (i,j) would have been unstable. Similarly, if student i prefers project center p to j and project center p prefers student i to s, then the initial match would have been unstable. This condition of stability that prevents any such blocking pairs is represented in the following pairwise elimination (PW) constraint set:

$$x_{sp} + x_{ij} \leq 1 \quad \forall\ s,i \in \mathcal{S},\ \forall\ p,j \in \mathcal{P} : ((j \succ_s p) \wedge (i \prec_j s)) \vee ((i \succ_p s) \wedge (j \prec_i p)). \tag{PW}$$

While intuitive, stability constraint set (PW) requires $\mathcal{O}(|\mathcal{S}|^2|\mathcal{P}|^2)$ constraints to ensure stability. In an effort to reduce the number of constraints, we propose two approaches to aggregate over i and j in (PW) that satisfy either the first *or* second set of blocking pair conditions. These are respectively introduced in constraint sets (SPC1) and (SPC2).

$$\sum_{\substack{j\in\mathcal{P}:\\ j\succ_s p}} \sum_{\substack{i\in\mathcal{S}:\\ i\prec_j s}} x_{ij} \leq \mathcal{M}_s(1 - x_{sp}) \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}. \tag{SPC1}$$

The constant $\mathcal{M}_s$ upper bounds the number of possible blocking pairs associated with student s. Following the first set of blocking pair conditions in (PW), constraint set (SPC1) ensures that if student s is assigned to project center p, then no pair (i,j) is matched where s prefers project center j to p and, concurrently, j prefers student s to i.

An alternative way to aggregate stability constraint set (PW) is

$$\sum_{\substack{i\in\mathcal{S}:\\ i\succ_p s}} \sum_{\substack{j\in\mathcal{P}:\\ j\prec_i p}} x_{ij} \leq \mathcal{M}_p(1 - x_{sp}) \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}. \tag{SPC2}$$

The constant $\mathcal{M}_p$ upper bounds the number of possible blocking pairs associated with project center p. Following the second set of blocking pair conditions in (PW), constraint set (SPC2) ensures that if student s is assigned to project center p, then no pair (i,j) is matched where p prefers student i to s and, concurrently, i prefers project center p

to j. Each of (SPC1) and (SPC2) prevents blocking pairs associated with justified envy. Therefore, combining either with (NWS) results in stable solutions with no blocking pairs associated with justified envy and waste, respectively.

Even with aggregation, the construction of constraint sets (SPC1) and (SPC2) can be time consuming, especially for large values of $|\mathcal{S}|$ and $|\mathcal{P}|$. We thus propose a recurrence relation to improve the construction time and computational performance of constraint sets (SPC1) and (SPC2) in Algorithms 1 and 2, respectively.

Enhancing the Generation of (SPC1) and (SPC2). Algorithm 1 generates stability constraint set (SPC1). Each student ranks project centers in their list in a descending order, then it iterates over each project center from most to least preferable to prevent blocking pairs associated to justified envy $(s, p^{(k)})$. That is, if student s is assigned to project center rank r, $p^{(r)}$, then there exists no assignment of a student i to project center $p^{(k)}$, $\forall\ k \in \{1, \ldots, r-1\}$, where s prefers $p^{(k)}$ more than $p^{(r)}$ and $p^{(k)}$ prefers s more than i. As a result, Algorithm 1 efficiently constructs constraints that prohibit blocking pairs associated with student s. Algorithm 2 generates stability constraint set (SPC2) in a manner similar to that of Algorithm 1. Each project center ranks students in its list in a descending order, then Algorithm 2 iterates over each student from most to least preferable to ensures that if project center p is assigned student $s^{(r)}$, then there exists no assignment of a project center j to student $s^{(k)}$ who p prefers more than $s^{(r)}$, and $s^{(k)}$ strictly prefers p to j, that is, it prevents blocking pairs $(s^{(k)}, p)$, $\forall\ k \in \{1, \ldots, r-1\}$. As a result, Algorithm 2 efficiently constructs constraints that prohibit blocking pairs associated with project center p.

The auxiliary variables α_{sp} and β_{sp} $\forall\ s \in \mathcal{S}$, $\forall\ p \in \mathcal{P}$ in Algorithms 1 and 2 are nonnegative integers and constructed over sorted preference lists. The value of $\mathcal{M}_s$ is no more than the number of project centers where student s prefers at least as much as the considering project center $p^{(r)}$ or $\sum_{k=1}^{r} |\mathcal{P}_s^{(k)}|$. The value of $\mathcal{M}_p$ is no more than the number of students that project center p prefers at least as much as the considering student $s^{(r)}$ or $\sum_{k=1}^{r} |\mathcal{S}_p^{(k)}|$. The bound of big-M values is the count of the possible blocking pairs that are available from inspecting the (incomplete) preference lists of s and p, respectively. We show in Theorem 5 of Appendix A that the recurrence relationship in Algorithms 1 and 2 preserves the stability constraint sets (SPC1) and (SPC2), respectively. A case illustrating Algorithms 1 and 2 can be found in Appendix C.

Algorithm 1: Recurrence relation for constructing (SPC1)

foreach $s \in \mathcal{S}$ **do**
  Rank $p \in \mathcal{P}$: $p^{(r)} \in \mathcal{P}_s^{(r)}, q_{sp^{(1)}} > q_{sp^{(2)}} > \dots$ // Order project centers in preference list of student s in nonincreasing fashion.
  for $r = 1$ **to** $|\mathcal{R}_p|$ **do**
    foreach $p^{(r)} \in \mathcal{P}_s^{(r)}$ **do**
      Define $\beta_{sp^{(r)}} = \sum_{i \in \mathcal{S}: i \prec_{p^{(r)}} s} x_{ip^{(r)}}$ // Sum over those students i that project center $p^{(r)}$ prefers strictly less than s.
      if *r=1* **then**
        Define $\alpha_{sp^{(1)}} = 0$ // There exists no blocking pair if student s is assigned to their first choice.
      else
        Define $\alpha_{sp^{(r)}} = \alpha_{sp^{(r-1)}} + \sum_{p \in \mathcal{P}_s^{(r-1)}} \beta_{sp}$ // If student s is assigned to $p^{(r)}$, then $\alpha_{sp^{(r)}}$ enumerates through project centers ranked better than r and prevents blocking pairs associated with s.
      Add a constraint: $\alpha_{sp^{(r)}} \leq \mathcal{M}_s(1 - x_{sp^{(r)}})$ // Add stability constraint (SPC1)

We have introduced four new many-to-one stability representations in the form of linear constraints – (VVM), (PW), (SPC1), and (SPC2) – along with a system of constraints (NWS) to eliminate blocking pairs associated with waste. In particular, (SPC1) and (SPC2) derive from (PW) and represent two new conceptual frameworks for establishing stability in many-to-one matching problems in a linear manner. Additional details on their modeling choices that are related to computational performance can be found in Appendix B. We also demonstrate that the new stability constraints achieve stability in the sense of removing justified envy and waste in Appendix A.

5. Computational Experiments

We now discuss experiments to compare and contrast the computational performance of different stability constraint representations. We conduct a variety of experiments on real-world and synthetic datasets on formulation (1) with stability constraint set (SPC1) as constructed by Algorithm 1, stability constraint set (SPC2) as constructed by Algorithm 2, stability constraint set (VVM), and stability constraint set (BBT). Notably, for the three new representations we use (NWS) to eliminate blocking pairs associated with waste, with the exception of the WPI datasets for SPC matching at Worcester Polytechnic Institute (WPI), in which the (NWS) constraints may be dropped because the objective only includes student preference in the utility function.

Algorithm 2: Recurrence relation for constructing (SPC2)

foreach $p \in \mathcal{P}$ **do**
  Rank $s \in \mathcal{S}$: $s^{(r)} \in \mathcal{S}_p^{(r)}, k_{s^{(1)}p} > k_{s^{(2)}p} > \dots$ // Order students in preference list of project center p in nonincreasing fashion.
  for $r = 1$ **to** $|\mathcal{R}_s|$ **do**
    foreach $s^{(r)} \in \mathcal{S}_p^{(r)}$ **do**
      Define $\beta_{s^{(r)}p} = \sum_{j \in \mathcal{P}: j \prec_{s^{(r)}} p} x_{s^{(r)}j}$ // Sum over those project centers j that student $s^{(r)}$ prefers strictly less than p
      if $r = 1$ **then**
        Define $\alpha_{s^{(1)}p} = 0$ // There exists no blocking pair if project center p is assigned its first choice.
      else
        Define $\alpha_{s^{(r)}p} = \alpha_{s^{(r-1)}p} + \sum_{s \in \mathcal{S}_p^{(r-1)}} \beta_{sp}$ // If center p is assigned student rank r, then $\alpha_{s^{(r)}p}$ enumerates through students ranked better than r and prevents blocking pairs associated with p.
      Add a constraint: $\alpha_{s^{(r)}p} \leq \mathcal{M}_p(1 - x_{s^{(r)}p})$ // Add stability constraint (SPC2)

5.1. Computational Setup and Datasets

All experiments were run using Gurobi Optimization 9.1 (2020) and Python, with up to 64 GB memory, under Red Hat Enterprise Linux version 7.2. Each model instance in the real datasets was run with the following Gurobi parameters: TimeLimit of 24 hours, OptimalityTol dual feasibility tolerance of 1E-9, and MIPGap optimality tolerance of 0. Our limited testing, detailed in Appendix B, results in a superior performance of using indicator constraints over Big-M conditions; hence, we model Big-M conditions in all stability representations using Gurobi indicator constraints. As we conducted extensive synthetic data experiments, each model instance was run with TimeLimit of one hour and MIPGap optimality tolerance of 1E-4.

We consider three datasets. The first is three years of student to project center matching data from WPI. The second dataset consists of 30 randomized HRT matching instances from Kwanashie and Manlove (2014). Finally, we synthetically generate a host of synthetic many-to-one matching problem instances, called "SynSPC". Detailed descriptions of the datasets are provided in Table 3.

5.2. Experimental Design

We evaluate the performance of (SPC1), (SPC2), (VVM), and (BBT) on the three datasets. Section 5.3 details experiments on three (annual) datasets concerning the preference lists

Table 3: Description of datasets used in our experiments.

Dataset	Source	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{R}_s\rvert$	$\lvert\mathcal{R}_p\rvert$	(NWS)	#Runs	#Replicates	TimeLimit
WPI	SPC matching at WPI	928; 927; 1126	46; 47; 57	$\leq \lvert\mathcal{S}\rvert$	3	✗	3	-	84,000 s
HRT	Kwanashie and Manlove (2014)	759	53	5	5-6	✓	1	30	3,600 s
SynSPC	Randomly Generated	Refer to Table 4				✓	210	10	3,600 s

of undergraduate students to be matched to worldwide project centers in the esteemed IQP program at WPI. In Section 5.3 we study how all variants of our stability representations perform across the three years for formulation (1), the stable matching with incomplete lists and ties and utility-weighted objectives.

In Section 5.4 we continue studying the performance of our stability representations by experimenting on the HRT dataset described in Section 5.1, and expand to also include a model adapted from Kwanashie and Manlove (2014) that we refer to as (`HRT1`). As a main difference of our work with that of Kwanashie and Manlove (2014) is our consideration of weighted utility, accordingly we augment the most compatible model of theirs, MAX-HRT formulations in their `model1`, with weighted project center utility. We conduct experiments on a moderately sized HRT dataset from Kwanashie and Manlove (2014) having 30 hospital-resident instances called *master list* and denoted "RDM-ML-1-5". Each instance contains 759 doctors, 53 hospitals, 775 available seats, and the ranking of the doctors, made based on their grades, distributed in the range of $[1, 5]$. While the preference of both doctors and hospitals in this dataset is incomplete, ties appear only in hospital preferences over doctors. The number of hospital preference tiers is five and the number of doctor preference tiers is either five or six. To compare our models with those of Kwanashie and Manlove (2014) using settings that are as similar as possible, we turn off their preprocessing with ties on the hospital side and convert their HRT instances to .lp files to solve using the Python API of Gurobi.

In Section 5.5 we perform a full factorial design by varying the parameters listed in Table 4 to generate a host of synthetic data instances, which we use to study the performance of each variant of stability representations. With this extensive set of experiments, we aim to thoroughly study the performance of our stability constraint representations across a wide variety of runs, in comparison with other leading stability constraint representations.

Given a single dataset with $|\mathcal{P}|$ project centers, each with an equal capacity c_p, we use the percentage of students $\%|\mathcal{S}|$ to compute the number of students, that is, $|\mathcal{S}| =$

$\lfloor |\mathcal{P}| \times c_p \times \%|\mathcal{S}| \rfloor$. The number of student preference tiers is the number of distinct ranks whereby students can place project centers, where placing project centers in the same tier indicates ties in their preference list. The `popularity` parameter indicates whether the popularity among project centers is either *uniform* if the likelihood that any student selects any project center into their preference list is equal across all project centers, or *nonuniform*, otherwise. We detail the generation of both popularity types in Appendix D. Without loss of generality, we assume r_{sp} represents the integer ordinal preference of student s for project center p, where $r_{sp} \in \{1, 2, \dots\}$, and that smaller values of r_{sp} indicate a more preferable project center for student s. Throughout our computational experiments, we define the cardinal preference value for student utility as $q_{sp} := \frac{1}{r_{sp}}$, which we note maintains the original order.

Table 4: The list of five parameters in the SynSPC dataset.

Parameter	Symbol	Levels
Number of Project Centers	$\lvert\mathcal{P}\rvert$	10, 30, 50
Project Center Capacity	c_p	10, 15, 20
Percentage of Students	$\%\lvert\mathcal{S}\rvert$	50%, 75%, 100%, 125%, 150%
Number of Student Preference Tiers	$\lvert\mathcal{R}_p\rvert$	$5, \dots, \max_{\leq \lvert\mathcal{P}\rvert}\{15, 25\}$
Popularity Among Project Centers	`popularity`	no, yes

We select the values of coefficients in the objective function of formulation (1) so that our models preemptively optimize each objective component in a strict ordering. For example, for the WPI datasets, we prioritize maximizing the number of student placements first, followed by student utility, and then project center utility. In general, each coefficient of the objective component is chosen in such a way that the contribution to the objective function of the considering component is strictly less than the minimum contribution from placing any single student of the component considered in an immediately higher order. We provide greater details on how we carefully derived coefficients used in the objective function of each dataset in Appendix E.

5.3. Experiments with WPI Datasets

Worcester Polytechnic Institute (WPI) is a private technological university in the Northeast United States that features a project-based and globally engaged curriculum. Perhaps the most distinctive program at WPI is the Interactive Qualifying Project (IQP), which

provides students an opportunity to apply their technical domain knowledge and skills to real-world projects. As part of the undergraduate degree requirement, WPI students have the opportunity to complete their IQP in one of approximately 50 off-campus project centers situated around the globe. These students go through a competitive selection process, followed by rigorous cultural preparation. Upon the arrival of students at each project center, students are formed into specific project teams. For many years the process of matching students to project centers was handled manually. Due to increasing complexity, from 2018 onward a version of our approach has been used to recommend the matching of students to off-campus project centers where a final decision is concluded only after careful review by project center directors and the program administrators.

The present process requires students to rate each project center in one of three tiers that are assigned student utility weights of 1, 0.5, and 0, respectively. In an effort to increase access, students are required to choose at least three project centers in the first tier and at least six options in the first and second tiers combined. This process generated WPI datasets for the academic years of 2017–2018, 2018–2019, and 2019–2020.

We set the utility u_{sp} in the objective function of formulation (1) to the summation of weighted student utility and weighted project center utility, $\gamma_1 q_{sp} + \gamma_2 k_{sp}$. The values of γ_1 and γ_2 are carefully chosen so that the model preemptively optimizes first maximizing student placement, second maximizing student utility, and finally maximizing project center utility in a strict order. We detail how we compute these coefficients in Appendix E.1. Table 5 reveals the result of stable SPC formulation (1) with varying stability representations across three years of the WPI datasets.[4]

By enforcing stability, all blocking pairs are eliminated. Across all three datasets, all four methods exhibit similar performance, as evidenced by the integrality gap which for all methods is below 2%. Of further note is that many methods found strong integer feasible solutions in times well earlier than the allotted run time, and the stable outcomes of all methods for the year 2018–2019 are solved to optimality. We thus conclude that all methods perform reasonably well. Some overall trends include that (`SPC2`) is consistently the sparsest representation of stability, both in the relative lack of nonzeros and density ratio[5].

[4] The complete result of the stable SPC formulation (1) when varying stability representations across three years of the WPI datasets is shown in Table 11 of Appendix F.

[5] Model density is defined as $\frac{\#\text{Nonzeros}}{\#\text{Rows} \times \#\text{Columns}}$

Table 5: Comparison of SPC models with various stability constraint representations over three different years of WPI data.

Year	**2017-2018**					**2018-2019**					**2019-2020**				
Metric	**Stability Constraint Type**					**Stability Constraint Type**					**Stability Constraint Type**				
	None	**SPC1**	**SPC2**	**VVM**	**BBT**	**None**	**SPC1**	**SPC2**	**VVM**	**BBT**	**None**	**SPC1**	**SPC2**	**VVM**	**BBT**
Objective Value	**928.98**	**927.96**	926.96	927.96	926.96	**927.99**	**927.99**	**927.99**	**927.99**	**927.99**	1,126.90	1,106.87	**1,107.86**	1,107.85	1,106.86
Best Bound	**928.98**	**927.96**	928.98	927.98	928.97	**927.99**	**927.99**	**927.99**	**927.99**	**927.99**	1,126.90	1,124.88	1,126.89	**1,124.58**	1,126.84
MIPGap (%)	**0**	**0.0003**	0.2172	0.0021	0.2167	**0**	**0**	**0**	**0**	**0**	0	1.6275	1.7173	**1.5098**	1.8051
Best Incumbent Time	0.2	75,264	28,319	5,763	43,313	0.11	46.26	**8.25**	97.41	78.17	0.14	86,400	85,305	31,707	86,400
Run Time	0.2	86,400	86,400	86,400	86,400	0.11	46.26	**8.25**	97.41	78.17	0.14	86,400	86,400	86,400	86,400
Total Students	928					927					1,126				
Total Capacity															
Tier-1 Placements	885	853	**863**	849	**863**	**927**	**927**	**927**	**927**	**927**	1,049	**988**	976	951	965
Tier-2 Placements	43	74	63	78	63	0	0	0	0	0	77	118	131	156	141
Unassigned	0	**1**	2	**1**	2	0	0	0	0	0	0	20	**19**	**19**	20

For the year 2018–2019, it is clear that while all methods perform reasonably well (perhaps due to the structure of the dataset), no approaches are superior to (`SPC2`) in terms of model build time, runtime, and shortest time to find the best incumbent. For the year 2017–2018, (`SPC1`) outperforms the other methods in terms of quality of solution, the smallest optimality gap, and total number of students placed. For the year 2019–2020, (`SPC2`) slightly outperforms the others in terms of solution quality, while being slightly outperformed by (`VVM`) in terms of optimality gap. In summary, all stability constraint sets perform comparatively well on the three WPI datasets. While the results were similar, (`SPC2`) outperforms the other stability representations in two out of three years, 2018–2019 and 2019–2020.

5.4. Comparing Performance of Stability Representations on the HRT Dataset

In light of the encouraging performance in Section 5.3, we consider the following which is an adjustment of our objective function (1a), where the utility u_{sp} is set to the project center utility k_{sp}, and the weight γ is chosen so that it first prioritizes maximizing student placement, followed by maximizing project center utility:

$$\text{maximize} \quad \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} x_{sp} + \gamma \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} k_{sp}x_{sp}. \tag{3}$$

Here, the weight γ is defined as $\frac{1}{|\mathcal{S}|+\varepsilon}$, where the value of ε is set to 1E$-$6, and the utilities k_{sp} are defined as the reciprocal of the tier for student s for project center p. Note the stability constraint set in the third formulation of Kwanashie and Manlove (2014) is (`HRT`) which is equivalent to (`BBT`) as discussed in Section 4.

Table 6 highlights the results of our experiments with the HRT dataset. In discussing its results, we aggregate the results of the 30 selected master-list instances by averaging

over the runtime for each stability representation. We summarize the results in Table 6; (SPC1) exhibits the best average runtime, followed by (VVM), (SPC2), (HRT1), and (BBT), respectively. Out of the 30 instances, 17 of (SPC2), 5 of (SPC1), 5 of (VVM), and 3 of (HRT1), respectively, result in the best runtime. The performance of (BBT) and (HRT1) are comparatively the same. This supports that, at least in a limited manner, our introduced stability constraints perform well in the real-world HRT instances that maximize stable matching with utility.

Table 6: Performance comparison of four stability constraint sets versus the HRT1 constraint set on the HRT dataset.

Method	SPC1	SPC2	VVM	BBT	HRT1
Average Runtime (s)	**628.65**	726.09	685.28	1,600.62	1,563.36
#Best Runtime (out of 30 instances)	5	**17**	5	-	3

Stability representations (SPC1), (SPC2), and (VVM) have exhibited competitive performance on the WPI and HRT datasets. We now pursue an in-depth investigation of their performance on the SynSPC dataset, to both confirm our understanding and investigate further properties of the data.

5.5. Comparing Performance of Stability Representations on SynSPC Datasets

We apply formulation (1) with the utility u_{sp} set to the weighted student utility in which the value of γ is chosen so that the model first prioritizes maximizing student placement, followed by student utility. The full factorial experiment over all levels yields 210 runs. We create 10 randomized data instances, or replicates, for each unique set of parameters (runs) to mitigate variation. Of these 210 runs, we remove 23 indeterminate runs in which the absolute difference in runtimes of all pairs of the 10 instances across all stability representations are within 0.5 seconds of one another. Of these excluded runs, approximately half contain instances that are solved to global optimality, while the remainder contain instances for which a feasible solution was not found within the time limit. This removal avoids diluting our results, thereby better differentiating the solver performance on the various stability representations. As a result, we focus our study on the remaining 187 runs (1,870 instances in total). We examine the performance of the four stability representations on both the 1,870 instances and the 187 runs. From these results we gain insights into the parameter settings under which each representation excels.

We divide each of the 1,870 instances according to one of three outcomes: solved to optimality, suboptimal (timed out), or no solution found within the time limit; each type

of instance is depicted in the first, second, and third segment, respectively, according to the vertical dotted gray lines in Figure 1 and the corresponding Table 7. The performance of the four methods is reported by the runtime if the solution is solved to optimality, and by the MIPGap otherwise. We also report average runtime and MIPGap (AvgTime and AvgMIPGap) and maximum runtime and MIPGap (MaxTime and MaxMIPGap) for each method where applicable in Table 7.

Figure 1 **(Left) Cumulative distribution plot comparing each stability constraint set on 1,870 instances. (Right) Close-up of the left segment of the cumulative distribution plot.**

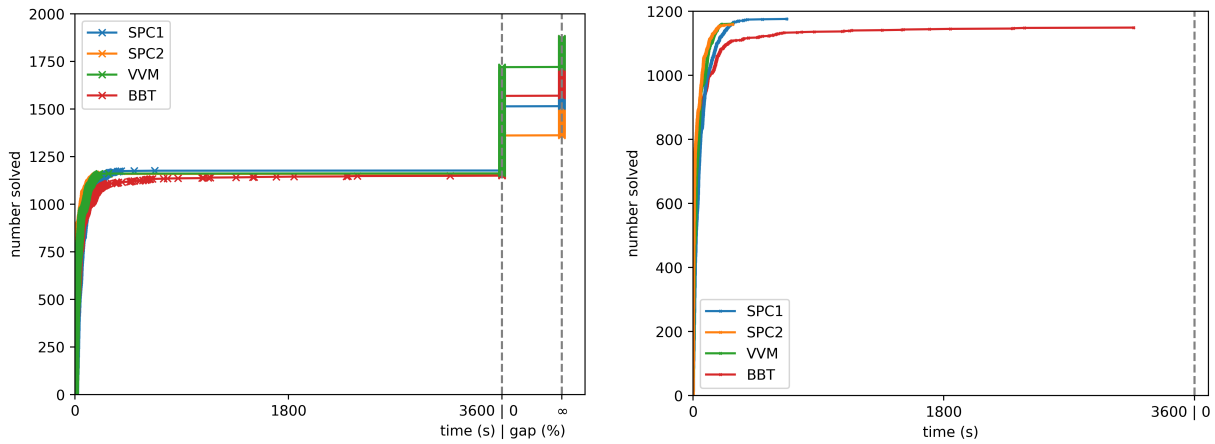

Table 7: Cumulative distribution of performance of the four stability representations.

	Runtime ≤ 3600 s			Timed out					
	Optimal			Suboptimal				No Solution Found	
Method	AvgTime	MaxTime	Count	AvgMIPGap	MaxMIPGap	Increment	Count	Increment	Count
SPC1	54.5	672.9	**1,176**	0.02%	0.06%	338	1,514	356	1,870
SPC2	**26.0**	285.5	1,159	0.04%	0.17%	202	1,361	509	1,870
VVM	35.2	**282.4**	1,161	**0.01%**	**0.03%**	**559**	**1,720**	**150**	1,870
BBT	71.7	3163.1	1,149	**0.01%**	**0.03%**	420	1,569	301	1,870

Based on Table 7, the first segment shows that all methods exhibit roughly similar performance with smaller average and maximum runtimes of all optimal instances, with perhaps (BBT) having a slightly less promising performance. It can also be seen that (SPC2) is somewhat superior because its instances result in the smallest average runtime and are solved to optimality within 300 seconds, while the instances of (VVM) also perform well, solving in under 300 seconds and having the smallest maximum runtime, and (SPC1) solves the greatest number of instances. The second segments show that, for instances

that are not solved to optimality, all methods exhibit roughly similar performance with average MIPGap values of just above zero. This implies that the suboptimal instances of all methods have virtually zero percent gap, but are not technically solved to optimality due to secondarily favoring some combination of utility. The timed-out instances of (`VVM`) find the greatest number of feasible solutions in this section. On the other hand, while (`SPC2`) outperforms others when the instances can be solved to optimality, it finds the fewest feasible solutions in this section. Finally, the third segment shows that (`SPC2`) is the method that has the most instances where no feasible solution was found within the time limit. We observe that instances for which (`SPC2`) timed out are primarily those that the number of students is greater than the total available capacity ($\%|\mathcal{S}| > 100\%$), whereas (`VVM`) is the method most likely to find some feasible solution, followed by (`BBT`), and (`SPC1`), respectively.

We now examine the performance of the four stability representations on the 187 runs. The number of runs where at least one instance out of 10 instances has a feasible solution (not necessarily solved to global optimality) of each method over all 187 runs is as follows: (`SPC1`): 169, (`SPC2`): 160, (`VVM`): **185**, and (`BBT`): 177. We categorize each of the 187 runs into three classes: (1) all 10 instances solved to global optimality, (2) some solved to global optimality and some timed out, and (3) all 10 instances timed out. Runtime is considered in the first two classes when some instances solved to global optimality and MIPGap is considered when all instances timed out (MIPGap $\geq$ 1E$-$4). Note that we omit instances where the method cannot find a feasible solution within the time limit from the ensuing discussion. Table 8 summarizes the best method with the lowest average runtime and MIPGap resulted from the 187 runs; (`SPC2`) performs the best when all instances are solved to global optimality, whereas (`VVM`) tends to perform best when some instances time out.

We now consider the effect of each parameter on the performance of each stability constraint set for the purpose of making recommendations for the best methods across various parameter settings and levels. Our initial exploratory data analysis shows that the most important parameters on the runtime are in the following order: $\%|\mathcal{S}|, |\mathcal{R}_p|, c_p, |\mathcal{P}|$, and `popularity`. All instances in Class 1 feature $\%|\mathcal{S}| \leq 100$. Class 2 includes instances with $\%|\mathcal{S}| \geq 100$ and a combination of small to large levels of $|\mathcal{P}|$ and c_p. On the other hand, Class 3 features instances with $\%|\mathcal{S}| > 100$ and medium to large levels of $|\mathcal{P}|$ and c_p. We analyze the data and identify general trends according to certain parameters.

Table 8: Count of each method resulting in the lowest average runtime and MIPGap for the three classes across all 187 runs.

	Metric	SPC1	SPC2	VVM	BBT	Total
Class 1: all 10 instances solved to optimality	Runtime	1	89	13	12	115
Class 2: some solved to optimality and some timed out	Runtime	0	1	42	4	47
Class 3: all 10 instances timed out	MIPGap	0	2	12	11	25
		1	92	67	27	187

The (SPC2) method clearly outperforms others in Class 1. Classes 2 and 3 feature larger levels of parameters that are more difficult to solve with $\%|\mathcal{S}| > 100$. The (VVM) method clearly outperforms other in Class 2, while (VVM) and (BBT) perform relatively better in Class 3. As the problem size tends to grow larger from Class 1 to Class 3, so does the computational complexity. Table 9 summarizes our findings by the range of parameter values whereby each method is expected to perform well. We note for the overlapping range of parameters when $\%|\mathcal{S}| > 100$ of (SPC1), (VVM), and (BBT). (VVM) and (BBT) have more advantage as their range includes larger value of $|\mathcal{R}_p|$. However, (VVM) outshines (BBT) when $\%|\mathcal{S}| > 125$.

Table 9: Summary of parameters where respective methods are expected to perform well.

Method	Range of Parameters
(SPC2)	$\%\|\mathcal{S}\| \leq 100$
(SPC1)	$100 \leq \%\|\mathcal{S}\| \leq 125$; $\|\mathcal{R}_p\| \leq 10$; $c_p \leq 15$; $\|\mathcal{P}\| \leq 40$
(BBT)	$100 \leq \%\|\mathcal{S}\| \leq 125$; $\|\mathcal{R}_p\| \leq 20$; $c_p \leq 15$; $\|\mathcal{P}\| \leq 40$
(VVM)	$100 \leq \%\|\mathcal{S}\| \leq 150$; $c_p \leq 15$

In summary, across all metrics, (SPC2) clearly outperforms the others when the number of students does not exceed the number of seats. However, the performance of (SPC2) may fairly quickly degrade when the number of students begins to exceed the capacity. All other methods have less of a clear demarcation. For particularly difficult instances, such as when $\%|S|$ is relatively high and large values of either $|\mathcal{P}|$ or c_p or both, it remains unclear under which parameters different methods outperform others. In this case, our result suggests that (VVM) is the most well-rounded method, while (BBT) and (SPC1) also perform well when all parameters are in moderate ranges.

6. Conclusion

We study many-to-one stable matching with ties and incomplete lists through the lens of an integer optimization. We introduce several new representations of stability and demonstrate

that they remove justified envy and waste, and conduct a comprehensive study on their computational performance on real and synthetic datasets. Our framework makes use of lexicographic optimization to construct an objective function that places a strict ordering on the priority of objective components. The first component maximizes the number of total matches, the second component maximizes the total weighted student preferences, with a possible final component maximizing the total project center side preferences.

Computational experiments on real-world datasets demonstrate that our new stability representations can generate quality matching outcomes in reasonable times. Specifically, computational experiments of stability representations on real-world datasets show that (`SPC1`), (`SPC2`), and (`VVM`) are promising in SPC matching and similar applications, such as school choice problems and hospital-residency matching. Computational experiments of stability representations on the selected *master list* instances from Kwanashie and Manlove (2014) show that when utility is considered in the objective function the performance of our proposed stability constraint sets outperform existing methods. Furthermore, our experiments demonstrate that (`SPC2`) is a promising method for typical real-world applications where the number of applicants does not exceed the number of seats, which can be seen in school choice problems and hospital-residency matching. In all cases, stability representation (`VVM`) exhibits solid performance.

A form of our optimization-based methods is being implemented at Worcester Polytechnic Institute (WPI), a private university in the Northeast United States, to assign students to global project centers via integer optimization. The lexicographic construction of our objective function allows the model to optimize each component of the objective function in the desired order, that is, total placement and utility. As a result, the matching outcomes incorporate student and project center director preferences while maintaining maximum student placement and ensuring stability.

While integer optimization approaches may require careful construction to be computationally competitive, they excel in terms of effortless addition of side constraints and additional aspects that other algorithmic approaches are unable to easily incorporate without a major redesign. The overlapping performance of (`SPC1`), (`VVM`) and (`BBT`) in our simulation experiment requires further exploration to clearly delineate recommendations for their respective use. Future targeted experiments on specific sets of parameters may

allow for larger time limits and greater information with respect to optimality. The formation of teams in the context of stable matching with incomplete lists and ties and utility-weighted objectives is another research avenue. Finally, further investigation on our disaggregate constraints (PW) is needed to determine how they could be further leveraged to obtain computational efficiencies.

Acknowledgements

We thank the Global Experience Office (GEO) for providing the WPI datasets; especially, Anne Ogilvie, Erin Bell, Deborah Fusaro, Dawn Farmer, and Kent Rissmiller, the program coordinators in the GEO, for organizing the SPC matching. We thank Mark Santiago, Alfred Scott, and Andrew Whalen for their technical support on the back-end implementation at WPI. We thank Camila Dias, Lin Jiang, and Elizabeth Karpinski, who initiated the initial optimization-based model for the WPI SPC matching. We also thank Dr. Adrienne Hall-Phillips who worked with Trapp and WPI graduate students Xin Liu and Xuechun Li, to conduct a full-scale pilot test of initial optimization-based model for the IQP matching. Lastly, we thank David Manlove, Maxence Delorme and Alex Teytelboym for their comments and suggestions that greatly improved this paper.

References

Abdulkadiroğlu A, Pathak P, Roth AE, Sönmez T (2006) Changing the Boston school choice mechanism. Technical report, National Bureau of Economic Research.

Abdulkadiroğlu A, Sönmez T (2003) School choice: A mechanism design approach. *American economic review* 93(3):729–747.

Ágoston KC, Biró P, McBride I (2016) Integer programming methods for special college admissions problems. *Journal of Combinatorial Optimization* 32(4):1371–1399.

Ahani N, Andersson T, Martinello A, Teytelboym A, Trapp AC (2021) Placement Optimization in Refugee Resettlement. *forthcoming in Operations Research* .

Alcalde J, Barberà S (1994) Top dominance and the possibility of strategy-proof stable solutions to matching problems. *Economic theory* 4(3):417–435.

Baïou M, Balinski M (2000) The stable admissions polytope. *Mathematical programming* 87(3):427–439.

Bertsimas D, King A, Mazumder R (2016) Best subset selection via a modern optimization lens. *The Annals of Statistics* 44(2):813–852, URL `http://www.jstor.org/stable/43818629`.

Delacrétaz D, Kominers SD, Teytelboym A (2019) Matching mechanisms for refugee resettlement. Technical report, University of Oxford, URL `http://t8el.com/wp-content/uploads/2019/12/DKT-MMRR-Dec2019.pdf`.

Delorme M, García S, Gondzio J, Kalcsics J, Manlove D, Pettersson W (2019) Mathematical models for stable matching problems with ties and incomplete lists. *European Journal of Operational Research* 277(2):426–441.

Gale D, Shapley LS (1962) College admissions and the stability of marriage. *The American Mathematical Monthly* 69(1):9–15.

Gurobi Optimization, LLC (2020) Gurobi Optimizer Reference Manual. Version 9.1. URL `http://www.gurobi.com`.

Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, del R'ıo JF, Wiebe M, Peterson P, G'erard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE (2020) Array programming with NumPy. *Nature* 585(7825):357–362, URL `http://dx.doi.org/10.1038/s41586-020-2649-2`.

Kojima F, Ünver MU (2014) The Boston school-choice mechanism: An axiomatic approach. *Economic Theory* 55(3):515–544.

Kwanashie A, Manlove DF (2014) An integer programming approach to the hospitals/residents problem with ties. *Operations Research Proceedings 2013*, 263–269 (Springer).

Maass KL, Lo VMH, Weiss A, Daskin MS (2015) Maximizing diversity in the engineering global leadership cultural families. *Interfaces* 45(4):293–304.

Manlove DF, O'Malley G, Prosser P, Unsworth C (2007) A constraint programming approach to the hospitals/residents problem. *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, 155–170 (Springer).

Roth AE (1982) The economics of matching: Stability and incentives. *Mathematics of operations research* 7(4):617–628.

Roth AE (1985) The college admissions problem is not equivalent to the marriage problem. *Journal of economic Theory* 36(2):277–288.

Roth AE (1986) On the allocation of residents to rural hospitals: a general property of two-sided matching markets. *Econometrica: Journal of the Econometric Society* 425–427.

Roth AE, Rothblum UG, Vande Vate JH (1993) Stable matchings, optimal assignments, and linear programming. *Mathematics of operations research* 18(4):803–828.

Roth AE, Sotomayor M (1989) The college admissions problem revisited. *Econometrica: Journal of the Econometric Society* 559–570.

Roth AE, Sotomayor M (1992) Two-sided matching. *Handbook of game theory with economic applications* 1:485–541.

Shapley L, Scarf H (1974) On cores and indivisibility. *Journal of mathematical economics* 1(1):23–37.

Shimada N, Yamazaki N, Takano Y (2020) Multi-objective optimization models for many-to-one matching problems. *Journal of Information Processing* 28:406–412, URL `http://dx.doi.org/10.2197/ipsjjip.28.406`.

Vande Vate JH (1989) Linear programming brings marital bliss. *Operations Research Letters* 8(3):147–153.

Wolsey LA (1998) *Integer programming* (John Wiley & Sons).

Appendix A: Theoretical Results Related to New Stability Representations

We present a sequence of theoretical results that demonstrate our proposed stability representations yield stable solutions. Recall that the many-to-one matching in this study is in the context of *HRT-W* in which the objective emphasizes maximizing the number of matches as well as weighted utilities for both sides of the matching, and the preference lists can be derived from ranking of corresponding weights assigned to each matching pair. Proposition 1, Theorem 1 and Theorem 2, respectively, show that, in the the context of *HRT-W*, the stability constraint sets (VVM), (SPC1), and (SPC2) in conjunction with (NWS) result in no blocking pairs associated with either justified envy or waste, resulting in stable solutions. Theorem 3 shows the equivalence of (BBT) and (VVM). Theorem 4 shows the equivalence of (SPC1), (SPC2), and (VVM). Finally, Theorem 5 shows that the recurrence relationship in Algorithms 1 and 2, respectively, preserves the stability constraint sets (SPC1) and (SPC2).

PROPOSITION 1. *The one-to-one stability constraint of Vande Vate (1989) can be extended to the HRT-W context as follows:*

$$\sum_{i \prec_p s} x_{ip} \leq c_p \left(1 - \sum_{j \prec_s p} x_{sj}\right) \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}, \tag{VVM}$$

and when combined with (NWS)*, ensures a stable system with no justified envy or waste.*

Proof. We translate constraint set (VV) to the many-to-one context: if student s is assigned to a project center less desirable than p $\left(\sum_{j \succeq_s p} x_{sj} = 0\right)$, then project center p may be filled with up to c_p students that it prefers at least as much as s $\left(\sum_{i \prec_p s} x_{ip} = 0\right)$. We obtain $\sum_{i \prec_p s} x_{ip} \leq c_p \sum_{j \succeq_s p} x_{sj}$ as a result.

All students are assigned to either a project center in their list or to the virtual project center, thus $\sum_{j \prec_s p} x_{sj} + \sum_{j \succeq_s p} x_{sj} = 1$. By substituting $\sum_{j \succeq_s p} x_{sj}$, we obtain $\sum_{i \prec_p s} x_{ip} \leq c_p \left(1 - \sum_{j \prec_s p} x_{sj}\right)$ for each $s \in \mathcal{S}$ and $p \in \mathcal{P}$ which is equivalent to constraint set (VVM).

Constraint set (VVM) eliminates blocking pairs (s, p) associated with justified envy in which project center p prefers student s to student i in its assignment and, concurrently, student s prefers project center p to their match j. Taken in conjunction with system (NWS) that forbids waste, this ensures there are no blocking pairs associated with either justified envy or waste, and thereby a stable outcome for (VVM). ∎

THEOREM 1. *A matching* $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}| \times |\mathcal{P}|}$ *is stable if and only if constraint sets* (SPC1) *and* (NWS) *are satisfied.*

Proof. First, we show that in the presence of (NWS), the outcomes of (SPC1) contain no blocking pair associated with waste. By contradiction, suppose (NWS) is violated, then there exists at least one empty seat at project center p where student i prefers p to their match and p prefers i to an empty seat. Recall that (NWa) activates z_p to 1 when there is at least one empty seat at project center p. When constraint set (NWb) is violated, the value on its left-hand side will be greater than zero, that is, $\sum_{i \in \mathcal{S}_p} \sum_{\substack{j \in \mathcal{P}: \\ j \prec_i p}} x_{ij} > 0$. Thus, there exists a blocking pair (i, p) in which student i is assigned to project center j who prefers p to j and, concurrently, project center p is willing to accept student i instead of having an empty seat; hence the outcome is not stable.

In the presence of (NWS), it is then sufficient to show that a matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{P}|}$ is stable if and only if constraint set (SPC1) is satisfied, that is it forbids blocking pairs associated with justified envy.

Sufficiency. By contradiction, suppose (SPC1) is violated. This implies the existence of both $(s,p) \in \mathcal{S}\times\mathcal{P} : x_{sp} = 1$ and $(i,j) \in \mathcal{S}\times\mathcal{P} : x_{ij} = 1$ with $j \succ_s p, i \prec_j s$. Thus, (s,j) forms a blocking pair associated with justified envy and matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{P}|}$ violates the notion of stability.

Necessity. Now, suppose a matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{P}|}$ is *not* stable. A blocking pair associated with a match $(s,p) \in \mathcal{S}\times\mathcal{P}$ is either (s,j) or (i,p). Without loss of generality, suppose a blocking pair is (s,j) in which student s is assigned to project center p and student i is assigned to project center j, but $j \succ_s p$ and $i \prec_j s$. This implies that $\exists (s,p) \in \mathcal{S}\times\mathcal{P} : x_{sp} = 1$ while $\sum_{\substack{j\in\mathcal{P}:\\ j\succ_s p}} \sum_{\substack{i\in\mathcal{S}:\\ i\prec_j s}} x_{ij} \geq 1$. Thus, (SPC1) is not satisfied. A similar argument exists for a blocking pair (i,p) where $(s,p) \in \mathcal{S}\times\mathcal{P}$ and $(i,j) \in \mathcal{S}\times\mathcal{P}$, but $p \succ_i j$ and $s \prec_p i$. ■

THEOREM 2. *A matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{P}|}$ is stable if and only if constraint sets* (SPC2) *and* (NWS) *are satisfied.*

Proof. First, we show that in the presence of (NWS), the outcomes of (SPC2) contain no blocking pair associated with waste. This can be shown in a similar manner as in Theorem 1. In the presence of (NWS), it is now sufficient to show that a matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{P}|}$ is stable if and only if constraint sets (SPC2) is satisfied, that is it forbids blocking pairs associated with justified envy.

Sufficiency. By contradiction, suppose (SPC2) is violated. This implies the existence of both $(s,p) \in \mathcal{S}\times\mathcal{P} : x_{sp} = 1$ and $(i,j) \in \mathcal{S}\times\mathcal{P} : x_{ij} = 1$ with $i \succ_p s, j \prec_i p$. Thus, (i,p) forms a blocking pair and matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{P}|}$ violates the notion of stability.

Necessity. Now suppose a matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{P}|}$ is *not* stable. A blocking pair associated with a match $(s,p) \in \mathcal{S}\times\mathcal{P}$ is either (s,j) or (i,p). Without loss of generality, suppose a blocking pair is (i,p) in which student s is assigned to project center p and student i is assigned to project center j, but $p \succ_i j$ and $s \prec_p i$. This implies that $\exists (s,p) \in \mathcal{S}\times\mathcal{P} : x_{sp} = 1$ while $\sum_{\substack{i\in\mathcal{S}:\\ s\prec_p i}} \sum_{\substack{j\in\mathcal{P}:\\ p\succ_i j}} x_{ij} \geq 1$. Thus, (SPC2) is not satisfied. A similar argument exists for a blocking pair (s,j) where $(s,p) \in \mathcal{S}\times\mathcal{P}$ and $(i,j) \in \mathcal{S}\times\mathcal{P}$, but $i \prec_j s$ and $j \succ_s p$. ■

We now demonstrate first the equivalence of (BBT) and (VVM) in Theorem 3, followed by the equivalence of (SPC1), (SPC2), and (VVM) in Theorem 4.

THEOREM 3. *Stability constraint set* (VVM)*, in conjunction with* (NWS)*, achieves equivalent stable outcomes to that of* (BBT)*.*

Proof. As (NWS) forbids blocking pairs associated to waste, we show that (VVM) which forbids blocking pairs associated to justified envy is equivalent to (BBT). For each $s \in \mathcal{S}$ and $p \in \mathcal{P}$:

$$
\begin{aligned}
& c_p x_{sp} + c_p \sum_{\substack{j\neq p:\\ j\succeq_s p}} x_{sj} + \sum_{\substack{i\neq s:\\ i\succeq_p s}} x_{ip} \geq c_p && \text{stability constraint (BBT)} \\
\iff\quad & c_p \sum_{j\succeq_s p} x_{sj} + \sum_{i\succeq_p s} x_{ip} \geq c_p && \text{stability constraint (HRT)} \\
\iff\quad & c_p\Big(1 - \sum_{j\prec_s p} x_{sj}\Big) \geq c_p - \sum_{i\succeq_p s} x_{ip} \geq \sum_{i\prec_p s} x_{ip} && \text{every student needs to be matched} \\
\iff\quad & c_p\Big(1 - \sum_{j\prec_s p} x_{sj}\Big) \geq \sum_{i\prec_p s} x_{ip} && \text{stability constraint (VVM)}
\end{aligned}
$$

■

THEOREM 4. *Stability constraint sets* (SPC1) *and* (VVM)*, each in conjunction with* (NWS)*, achieve equivalent stable outcomes.*

Proof. Given that (NWS) is applied to both constraint sets, we prove by contradiction the equivalence of (SPC1) and (VVM) in which they forbids blocking pairs associated to justified envy.

Suppose (SPC1) holds, but (VVM) is violated. This implies that $\sum_{j \prec_{sp}} x_{sj} = 1$ and yet $\sum_{i \prec_{ps}} x_{ip} \geq 1$. Therefore, $\exists (s,j),(i,p) \in \mathcal{S} \times \mathcal{P} : x_{sj} = 1$ and $x_{ip} = 1$, where $j \prec_s p$ and $i \prec_p s$. Accordingly, x_{sp} is a blocking pair associated with justified envy, and thus stability constraint set (SPC1) cannot hold.

Now, suppose (VVM) holds, but (SPC1) is violated. This implies that $\exists (s,j),(i,p) \in \mathcal{S} \times \mathcal{P} : x_{sp} = 1$ and $x_{ij} = 1$, where $j \succ_s p$ and $i \prec_j s$. Accordingly, x_{sj} is a blocking pair associated with justified envy, and thus stability constraint set (VVM) cannot hold. ∎

The equivalence of (SPC2) and (VVM) can be shown in a similar manner, this proving the equivalence of (SPC1), (SPC2), and (VVM). Theorems 3 and 4 together imply the equivalence of (SPC1), (SPC2), (VVM), and (BBT). We now show that Algorithms 1 and 2 preserve stability constraint sets (SPC1) and (SPC2), respectively.

THEOREM 5. *The recurrence relation of the variables α_{sp} in Algorithm 1 preserves stability constraint sets* (SPC1)*.*

Proof. First, rank order the project centers $p \in \mathcal{P}$ according to the preference list of student s as $p^{(1)}, p^{(2)}, \ldots$; then, for any $s \in \mathcal{S}$ and $p^{(r)} \in \mathcal{P}$:

$$
\begin{array}{rrcll}
 & \alpha_{sp^{(r)}} & \leq & \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) & \\
\iff & \alpha_{sp^{(r-1)}} + \beta_{sp^{(r-1)}} & \leq & \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) & \\
\iff & \alpha_{sp^{(r-2)}} + \sum_{k=1}^{2} \beta_{sp^{(r-k)}} & \leq & \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) & \\
\iff & & \cdots & & \\
\iff & \alpha_{sp^{(1)}} + \sum_{k=1}^{r-1} \beta_{sp^{(r-k)}} & \leq & \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) & \\
\iff & \sum_{k=1}^{r-1} \beta_{sp^{(r-k)}} & \leq & \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) & \text{by definition } \alpha_{sp^{(1)}} = 0 \\
\iff & \sum_{k=1}^{r-1} \sum_{i \prec_{p^{(r-k)}} s} x_{ip^{(r-k)}} & \leq & \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) & \text{by definition } \beta_{sp^{(r)}} = \sum_{\substack{i \in \mathcal{S}: \\ i \prec_{p^{(r)}} s}} x_{ip^{(r)}} \\
\iff & \sum_{\substack{j \in \mathcal{P}: \\ j \succ_s p^{(r)}}} \sum_{\substack{i \in \mathcal{S}: \\ i \prec_j s}} x_{ij} & \leq & \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) & \text{by substituting } \{j \in \mathcal{P} : j \succ_s p^{(r)}\}.
\end{array}
$$

∎

A similar proof shows that the recurrence relation of the variable α_{sp} in Algorithm 2 preserves stability constraint set (SPC2).

Appendix B: Summary of Limited Computational Testing for Configuring Experiments

We computationally investigate some parameters to shed light on desirable configurations for our computational experiments. First, limited computational testing revealed that the constraint set (PW) tends to be outperformed by both of the aggregate versions (SPC1) and (SPC2). This was surprising given the common understanding that the performance of aggregate constraints is typically dominated by disaggregated constraints in terms of the linear programming relaxation solution quality (see, e.g., Wolsey 1998). Algorithms 1 and 2 use recurrence relationships to construct stability constraint sets (SPC1) and (SPC2) over sorted preference lists. As a result, the summation of binary variables appearing in (SPC1) and (SPC2) is not composed of binary variables, but rather an integer variable that is set equal to the sum of the respective binary variables.

Limited computational testing reveals that models with these integer variables that represent binary variable aggregations result in superior performance, over models with only binary variables.

Additionally, we performed a number of limited computational experiments to understand what parameters and settings resulted in the best performance. Specifically, we compared the performance between using Big-M conditions and Gurobi indicator constraints. Even though we reduced the magnitude of Big-M parameters that appear in stability constraint sets (SPC1) and (SPC2) by enumerating through binary variables x_{ij} representing possible blocking pairs related to justified envy which locating on the left-hand side of the constraint sets.

Appendix C: An Example to Demonstrate the Generation of Constraint Sets via Algorithm 1 and Algorithm 2

EXAMPLE 1. Suppose there are two project centers p_1, p_2, each with two seats, and three students s_1, s_2, s_3 with incomplete preference lists and ties. Preferences and priorities are shown in Figure 2.

Figure 2 **Illustrating SPC assignment with incomplete preference lists and ties of Example 1. Unranked project centers / students are unlisted, and entities ranked at the same level are enclosed within same brackets.**

$$\begin{aligned} s_1 &: (p_1) & p_1 &: (s_2, s_3), (s_1) \\ s_2 &: (p_1, p_2) & p_2 &: (s_1), (s_2), (s_3) \\ s_3 &: (p_2), (p_1) \end{aligned}$$

Algorithm 1 iterates over the preference of students, whereas Algorithm 2 iterates over the preference of project centers, both sorted in a descending order. While both algorithms are based on (PW), each results in a unique constraint set as shown in Table 10. We now illustrate the recurrence relations of Algorithms 1 and 2 for constructing (SPC1) and (SPC2).

Algorithm 1. Considering the sorted project centers of each student s in a descending order, Algorithm 1 creates constraints to prevent any assignments of other students who each sorted project center p prefers less than s. A different subset of these constraints is activated when student s is assigned to each sorted project centers $p \in \mathcal{P}$.

From the preference order of s_1, as s_1 prefers being unmatched to being assigned to p_2, the order of project centers of s_1 is $(p_1), (p_0), (p_2)$. Because p_1 prefers no one less than s_1, this generates $\beta_{1,1} = 0$ and $\alpha_{1,1} = 0$. Next, p_0 prefers everyone equally generates $\beta_{1,0} = 0$ and $\alpha_{1,0} = \alpha_{1,1} + \beta_{1,1}$. Then, p_2 prefers s_2 and s_3 less than s_1 generates $\beta_{1,2} = x_{2,2} + x_{3,2}$ and $\alpha_{1,2} = \alpha_{1,0} + \beta_{1,0}$.

From the preference order of s_2 where p_1 and p_2 have the same ranking, p_1 prefers s_1 less than s_2 generates $\beta_{2,1} = x_{1,1}$ and $\alpha_{2,1} = 0$, whereas p_2 prefers s_3 less than s_2 generates $\beta_{2,2} = x_{3,2}$ and $\alpha_{2,2} = 0$. Then, p_0 generates $\beta_{2,0} = 0$ and $\alpha_{2,0} = \alpha_{2,2} + \beta_{2,2} + \beta_{2,1}$.

From the preference order of s_3, first, p_2 prefers no one less than s_3 generates $\beta_{3,2} = 0$ and $\alpha_{3,2} = 0$. Next, p_1 prefers s_1 less than s_3 generates $\beta_{3,1} = x_{1,1}$ and $\alpha_{3,1} = \alpha_{3,2} + \beta_{3,2}$. Then, p_0 generates $\beta_{3,0} = 0$ and $\alpha_{3,0} = \alpha_{3,1} + \beta_{3,1}$.

All in all, Algorithm 1 generates 9 constraints of the form $\alpha_{sp} \leq \mathcal{M}_s(1 - x_{sp})$, the last of which is $\alpha_{3,0} = \alpha_{3,1} + \beta_{3,1} = x_{1,1} \leq \mathcal{M}_s(1 - x_{3,0})$.

Algorithm 2. Considering the sorted students of each project center p in a descending order, Algorithm 2 creates constraints to prevent any assignments of each sorted student s to any of other project centers where s prefers less than p. A different subset of these constraints is activated when project center p is assigned each sorted student $s \in \mathcal{S}$.

From the preference order of p_1 where s_2 and s_3 have the same ranking, s_2 prefers p_0 less than p_1 generates $\beta_{2,1} = x_{2,0}$ and $\alpha_{2,1} = 0$, whereas s_3 prefers p_0 less than p_1 generates $\beta_{3,1} = x_{3,0}$ and $\alpha_{3,1} = 0$. Next, s_1 prefers p_0 and p_2 less than p_1 generates $\beta_{1,1} = x_{1,0} + x_{1,2}$ and $\alpha_{1,1} = \alpha_{3,1} + \beta_{3,1} + \beta_{2,1}$.

From the preference order of p_2, first, s_1 prefers nowhere less than p_2 generates $\beta_{1,2} = 0$ and $\alpha_{1,2} = 0$. Next, s_2 prefers p_0 less than p_2 generates $\beta_{2,2} = x_{2,0}$ and $\alpha_{2,2} = \alpha_{1,2} + \beta_{1,2}$. Then, s_3 prefers p_1 and p_0 less than p_2 generates $\beta_{3,2} = x_{3,1} + x_{3,0}$ and $\alpha_{3,2} = \alpha_{2,2} + \beta_{2,2}$.

All in all, Algorithm 2 generates 6 constraints of the form $\alpha_{sp} \leq \mathcal{M}_p(1 - x_{sp})$, the last of which is $\alpha_{3,2} = \alpha_{2,2} + \beta_{2,2} = x_{2,0} \leq \mathcal{M}_s(1 - x_{3,2})$.

Note that we use Gurobi indicator constraints to model Big-M conditions in stability representations (`SPC1`) and (`SPC2`) as our limited computational testing shows their superior performance over Big-M conditions. We summarize constraint sets generated by Algorithm 1 and Algorithm 2 in Table 10.

Table 10: The difference in generation order, and constraints, as created by Algorithm 1 and Algorithm 2.

Left-hand Side of Constraints Generated by:	
Algorithm 1	Algorithm 2
$\beta_{1,1} = 0;\ \alpha_{1,1} = 0$	$\beta_{3,1} = x_{3,0};\ \alpha_{3,1} = 0$
$\beta_{1,0} = 0;\ \alpha_{1,0} = \beta_{1,1} + \alpha_{1,1}$	$\beta_{2,1} = x_{2,0};\ \alpha_{2,1} = 0$
$\beta_{1,2} = x_{2,2} + x_{3,2};\ \alpha_{1,2} = \beta_{1,0} + \alpha_{1,0}$	$\beta_{1,1} = x_{1,2} + x_{1,0};\ \alpha_{1,1} = \beta_{3,1} + \alpha_{3,1} + \beta_{2,1}$
$\beta_{2,2} = x_{3,2};\ \alpha_{2,2} = 0$	$\beta_{1,2} = 0;\ \alpha_{1,2} = 0$
$\beta_{2,1} = x_{1,1};\ \alpha_{2,1} = 0$	$\beta_{2,2} = x_{2,0};\ \alpha_{2,2} = \beta_{1,2} + \alpha_{1,2}$
$\beta_{2,0} = 0;\ \alpha_{2,0} = \beta_{2,2} + \alpha_{2,2} + \beta_{2,1}$	$\beta_{3,2} = x_{3,1} + x_{3,0};\ \alpha_{3,2} = \beta_{2,2} + \alpha_{2,2}$
$\beta_{3,2} = 0;\ \alpha_{3,2} = 0$	$\beta_{3,0} = 0;\ \alpha_{3,0} = 0$
$\beta_{3,1} = x_{1,1};\ \alpha_{3,1} = \beta_{3,2} + \alpha_{3,2}$	$\beta_{2,0} = 0;\ \alpha_{2,0} = 0$
$\beta_{3,0} = 0;\ \alpha_{3,0} = \beta_{3,1} + \alpha_{3,1}$	$\beta_{1,0} = x_{1,2};\ \alpha_{1,0} = 0$
Building (`SPC1`) via indicator constraints	Building (`SPC2`) via indicator constraints
$x_{1,1} = 1 \rightarrow \alpha_{1,1} = 0$	$x_{3,1} = 1 \rightarrow \alpha_{3,1} = 0$
$x_{1,0} = 1 \rightarrow \alpha_{1,0} = 0$	$x_{2,1} = 1 \rightarrow \alpha_{2,1} = 0$
$x_{1,2} = 1 \rightarrow \alpha_{1,2} = 0$	$x_{1,1} = 1 \rightarrow \alpha_{1,1} = 0$
$x_{2,2} = 1 \rightarrow \alpha_{2,2} = 0$	$x_{1,2} = 1 \rightarrow \alpha_{1,2} = 0$
$x_{2,1} = 1 \rightarrow \alpha_{2,1} = 0$	$x_{2,2} = 1 \rightarrow \alpha_{2,2} = 0$
$x_{2,0} = 1 \rightarrow \alpha_{2,0} = 0$	$x_{3,2} = 1 \rightarrow \alpha_{3,2} = 0$
$x_{3,2} = 1 \rightarrow \alpha_{3,2} = 0$	$x_{3,0} = 1 \rightarrow \alpha_{3,0} = 0$
$x_{3,1} = 1 \rightarrow \alpha_{3,1} = 0$	$x_{2,0} = 1 \rightarrow \alpha_{2,0} = 0$
$x_{3,0} = 1 \rightarrow \alpha_{3,0} = 0$	$x_{1,0} = 1 \rightarrow \alpha_{1,0} = 0$

Appendix D: Generating Student Preference List Based on the Two Types of `popularity` for SynSPC Dataset Parameter

The preference lists in our datasets are incomplete. Unlike the studies of Kwanashie and Manlove (2014) and Delorme et al. (2019) where the data in the randomly generated instances appears to be generated with ties on the hospital side only, we allow ties on both student and project center sides, with a greater tie density in student preference lists. We note that in the generated test instances of Delorme et al. (2019), all hospitals in the preference list of each doctor rank the particular doctor in the same tier. Additionally, the distribution of doctor grades is controlled by a skewness parameter, in which a skewness value of κ means that the most common doctor score is likely to occur κ times more than the least common, whereas the distribution of our student preference list is controlled by the popularity of each project center, according to the following interpretation. In the SynSPC dataset, we generate a discrete probability distribution for tier selection based on the popularity ranking of each project center, such that more popular project centers will have greater likelihood of being placed in the first tier of student preference lists.

The preference lists of students and project centers in Section 5.5 are generated to reflect the SPC matching in which student preferences are constructed in preference tiers where project centers in the same tier represent ties, and project center preferences are calculated based on a variety of observable student characteristics such as resume, transcript, language skill, and personal statement.

The student preferences is constructed based on the project center popularity where its distribution can be uniform or nonuniform. A *uniform* popularity indicates a uniform popularity distribution across all project centers, where each project center is likely to be selected by a student with a probability of $\frac{1}{|\mathcal{P}|}$. The chances that each project center will be selected in the first tier of student preference lists are equal. A *nonuniform* popularity indicates a nonuniform popularity distribution across all project centers, where more popular project centers are more likely to be selected to higher tier of student preference lists.

We define a discrete probability distribution of tier selection for project popularity rank $p \in \{1, 2, \dots\}$ as

$$\Pr(\text{rank} = p, \text{tier} = r) = \frac{|\mathcal{R}_p| - r + p}{|\mathcal{R}_p| \left(\frac{|\mathcal{R}_p|-1}{2} + p\right)} \quad \text{for preference tier } r \in \{1, \dots, |\mathcal{R}_p|\}.$$

For example, when $|\mathcal{R}_p| = 3$, the probability that the most popular project center (rank=1) is selected in the first, second, and third tier of student preference lists are $\frac{3}{6}$, $\frac{2}{6}$, and $\frac{1}{6}$, respectively; the probability that the second most popular project center (rank=2) is selected in the first, second, and third tier of student preference lists are $\frac{4}{9}$, $\frac{3}{9}$, and $\frac{2}{9}$, respectively; and so on. The scale difference between each project popularity rank can be adjusted by changing the scale of p values.

The probability distribution of the tier selection of a more popular project center results in higher chance that it will be selected in higher tier. The random selection of project popularity and preference lists is implemented using Python function `choice` in `numpy.random` library (Harris et al. 2020). With the use of this function, students place each project center to a tier according to a list of probability values generated from the corresponding project popularity rank. We prohibit student preference lists with empty first tier. Incomplete preference lists occur when students do not place project centers in every tier. Since we work with cardinal preferences, a utility value of a project center placed in the r^{th} tier is defined to be $\frac{1}{r}$ for $r = 1, \dots, |\mathcal{R}_p| - 1$, and zero if it is placed in the last tier, $r = |\mathcal{R}_p|$.

Appendix E: Selecting Coefficients of the Lexicographic Objective Functions

E.1. Objective Coefficients for the Stable SPC Formulation (1) of the WPI Datasets

To study the effect of different stability representations, we apply formulation (1) with the utility u_{sp} set to the summation of weighted student utility and weighted project center utility, $\gamma_1 q_{sp} + \gamma_2 k_{sp}$. We choose the values of γ_1 and γ_2 so that the objective function:

$$\text{maximize} \quad \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} x_{sp} + \gamma_1 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} q_{sp}x_{sp} + \gamma_2 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} k_{sp}x_{sp},$$

preemptively optimizes in the following order: i) maximize student placement, ii) maximize student utility, and iii) maximize project center utility.

First, we would like to ensure that the contribution to the objective function of total project center utility is strictly less than the minimum contribution of the student utility from placing any single student: $\gamma_2 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} k_{sp}x_{sp} < \gamma_1 \min_{\{s\in\mathcal{S},p\in\mathcal{P}\}} q^+_{sp}$. Second, we would like to ensure that the contribution to the objective function of total student utility is strictly less than the minimum contribution of a placement of any single student: $\gamma_1 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} q_{sp}x_{sp} < 1$. Finally, we would like to ensure that the contribution to the objective function of total student utility and total project center utility together is strictly less than the minimum contribution of a placement of any single student: $\gamma_1 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} q_{sp}x_{sp} + \gamma_2 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} k_{sp}x_{sp} < 1$. The last condition ensures that we obtain an objective value that reflects the number of student placements plus a small quantity of desired matching conditions. To do so, it is helpful to work backwards, so that first the γ_2 component is determined in term of γ_1, followed by solving for the value of the γ_1 component.

As a result, we set the value of γ_1 to be the reciprocal of the minimum of the total capacity and the number of students, plus the smallest possible positive contribution from student utility, plus a small positive value, that is $\gamma_1 = \frac{1}{\min_{\{s\in\mathcal{S},p\in\mathcal{P}\}} q^+_{sp} + \min\{\sum_{p\in\mathcal{P}} c_p, |\mathcal{S}|\} + \varepsilon}$. We set the value of γ_2 to be the product of γ_1 and the ratio of the minimum student utility of a placement of any single student to the minimum of the total capacity and the number of students plus a small positive value, $\gamma_2 = \frac{\min_{\{s\in\mathcal{S},p\in\mathcal{P}\}} q^+_{sp}}{\min\{\sum_{p\in\mathcal{P}} c_p, |\mathcal{S}|\} + \varepsilon}\gamma_1$.

For the WPI dataset, the capacity is dropped to its observed level in every year, $\sum_{p\in\mathcal{P}} c_p = |\mathcal{S}|$, and we prohibit the placement of students to unpreferred project centers in which $q_{sp} = 0$ via constraint (1d), which results in $\min_{\{s\in\mathcal{S},p\in\mathcal{P}\}} q^+_{sp} = 0.5$. Hence, $\gamma_1 = \frac{1}{0.5 + \sum_{p\in\mathcal{P}} c_p + \varepsilon}$ and $\gamma_2 = \frac{0.5}{\sum_{p\in\mathcal{P}} c_p + \varepsilon}\frac{1}{0.5 + \sum_{p\in\mathcal{P}} c_p + \varepsilon}$, where we set the value of ε to be 1E−6.

E.2. Objective Coefficients for SynSPC Dataset in Section 5.5

We apply formulation (1) with the utility u_{sp} set to the weighted student utility in which the value of γ is chosen so that the model first prioritizes maximizing student placement, followed by maximizing student utility. We set the value of γ to be the reciprocal of the minimum of the total capacity or the total number of students plus a small positive value, $\frac{1}{min\{\sum_{p\in\mathcal{P}} c_p, |\mathcal{S}|\} + \varepsilon}$; we set ε=1E-6. In this way, we ensure that $\gamma \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} q_{sp} < 1$, that is, the contribution to the objective function of placing any single student outweighs the benefit of all student utility values combined.

Appendix F: Additional Experimental Results

Table 11 shows the result of stable SPC formulation (1) with different stability constraint sets across three years of the WPI datasets. Note that all runs timed out at 24 hours for the first and last years, while all runs solved well under the time limit for the second year, this is due to different flexibility of students in choosing project centers in each year. In particular, we observe that students are more restrictive in their selection of project centers in the first and last years; as the result, the process is more competitive and it poses more difficulty in solving the models.

Table 11: Comparison of SPC models with different stability constraint sets on three WPI datasets.

Year	**2017-2018**					**2018-2019**					**2019-2020**				
Metric	**Stability Constraint Type**					**Stability Constraint Type**					**Stability Constraint Type**				
	None	**SPC1**	**SPC2**	**VVM**	**BBT**	**None**	**SPC1**	**SPC2**	**VVM**	**BBT**	**None**	**SPC1**	**SPC2**	**VVM**	**BBT**
Objective Value	**928.98**	**927.96**	926.96	927.96	926.96	**927.99**	**927.99**	**927.99**	**927.99**	**927.99**	1,126.90	1,106.87	**1,107.86**	1,107.85	1,106.86
Best Bound	**928.98**	**927.96**	928.98	927.98	928.97	**927.99**	**927.99**	**927.99**	**927.99**	**927.99**	1,126.90	1,124.88	1,126.89	**1,124.58**	1,126.84
MIPGap (%)	**0**	**0.0003**	0.2172	0.0021	0.2167	**0**	**0**	**0**	**0**	**0**	0	1.6275	1.7173	**1.5098**	1.8051
After Presolve															
#Constraints (Rows)	966	13,835	20,145	15,564	10,979	974	12,007	15,818	12,130	7,641	1,183	12,583	16,304	13,630	8,355
#Variables (Columns)	14,359	18,280	24,230	15,307	12,397	11,169	15,407	18,199	12,137	8,970	12,597	16,560	18,298	13,796	10,371
#Nonzeros	28,710	158,553	95,178	1,896,966	2,026,000	22,338	107,719	74,355	1,314,575	1,080,987	25,194	84,965	77,151	1,441,600	1,360,148
Model Density	0.0021	0.0006	**0.0002**	0.0080	0.0149	0.0021	0.0006	**0.0003**	0.0089	0.0158	0.0017	0.0004	**0.0003**	0.0077	0.0157
Total Students	928					927					1,126				
Total Capacity															
Tier-1 Placements	885	853	**863**	849	**863**	**927**	**927**	**927**	**927**	**927**	1,049	**988**	976	951	965
Tier-2 Placements	43	74	63	78	63	0	0	0	0	0	77	118	131	156	141
Unassigned	0	**1**	2	**1**	2	0	0	0	0	0	0	20	**19**	**19**	20
Blocking Pairs	150	0	0	0	0	0	0	0	0	0	293	0	0	0	0
Time to Build Model (s)	1.44	147.36	19.88	41.79	52.49	1.28	110.07	**20.04**	47.60	50.89	2.20	206.68	38.95	85.71	77.05
First Incumbent Time (s)	0.2	89	8,554	125	33,326	0.11	46.26	**8.25**	97.41	78.17	0.14	65	10,535	540	30,699
Best Incumbent Time (s)	0.2	75,264	28,319	5,763	43,313	0.11	46.26	**8.25**	97.41	78.17	0.14	86,400	85,305	31,707	86,400
Run Time (s)	0.2	86,400	86,400	86,400	86,400	0.11	46.26	**8.25**	97.41	78.17	0.14	86,400	86,400	86,400	86,400