3-23-2017

A Scenario-Based Parametric Analysis of Stable Marriage Approaches to the Army Officer Assignment Problem

Matthew D. Ferguson

A Scenario-Based Parametric Analysis of Stable Marriage Approaches to the Army Officer Assignment Problem

THESIS

Matthew D. Ferguson, MAJ, USA

AFIT-ENS-MS-17-M-128

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-17-M-128

A SCENARIO-BASED PARAMETRIC ANALYSIS OF STABLE MARRIAGE APPROACHES TO THE ARMY OFFICER ASSIGNMENT PROBLEM

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Matthew D. Ferguson, B.S., M.A.

MAJ, USA

March 2017

AFIT-ENS-MS-17-M-128

A SCENARIO-BASED PARAMETRIC ANALYSIS OF STABLE MARRIAGE APPROACHES TO THE ARMY OFFICER ASSIGNMENT PROBLEM

THESIS

Matthew D. Ferguson, B.S., M.A.
MAJ, USA

Committee Membership:

Dr. R. R. Hill
Chair

Dr. B. J. Lunday
Member

AFIT-ENS-MS-17-M-128

Abstract

This paper compares linear programming and stable marriage approaches to the assignment problem under conditions of uncertainty. Robust solutions should exhibit reduced variability in the presence of one or more additional constraints.

Several variations of each approach are compared with respect to solution quality, as measured by the overall social welfare among Officers and Assignments, and robustness as measured by the number of changes after a number of randomized perturbations. We examine the contrasts between these methods in the context of assigning Army Officers among a set of identified assignments. Additional constraints are modeled after realistic scenarios faced by Army assignment managers, with parameters randomized.

The Pareto efficient approaches, relative to these measures of quality and robustness, are identified and subjected to a regression analysis. The coefficients of these models provide insight into the impact the different scenarios under study, as well as inform any trade-off decisions between Pareto-optimal approaches.

AFIT-ENS-MS-17-M-128

To my Family, whose steadfast support has always sustained me.

To my God, with Whom all things are possible.

And to my Colleagues, whose camaraderie pulled me through to the end.

Acknowledgements

If I have seen further, it is by standing on the shoulders of giants.

– Sir Isaac Newton

I would like to express my gratitude to my faculty research advisor, Dr. Ray Hill, for all of his guidance, patience and assistance in navigating a bumpy road. And to Dr. Brian Lunday, who ignited my interest in optimization, thank you. His passion for our profession is an inspiration.

I also wish to thank CPT Nick Paul of the United States Army Human Resources Command for assisting in this work and providing a realistic context for this study.

Matthew D. Ferguson

Table of Contents

Page

List of Figures

List of Tables

A SCENARIO-BASED PARAMETRIC ANALYSIS OF STABLE MARRIAGE APPROACHES TO THE ARMY OFFICER ASSIGNMENT PROBLEM

I. Introduction

> *Every day, HRC executes distribution, strategic talent management, personnel programs and services Army wide in order to optimize Total Force personnel readiness and strengthen an agile and adaptive Army.*
>
> –Mission of the United States Army Human Resources Command [1]

1.1 Background

One of the missions of the United States Army's Human Resources Command (HRC) is to annually manage the assignment and distribution of 92,627 active duty Officers and 378,193 active duty Enlisted personnel to Army and Joint organization around the world. While most personnel are assigned to a large unit or geographic location for a period between three to four years, military necessity requires the periodic balancing of national security and professional development requirements with the personal desires of Soldiers and their Families. [2]

Army Officer assignments may be difficult to address, given the many concerns surrounding assignments, both from the perspective of the Army and from Officers. While current practice involves Officers submitting personal preferences, ordinal listings communicating their most desirable assignments, assignment managers at HRC manually develop assignment *slates* or matchings of Officers to assignments.

The Army invests heavily in the training, education and experiences of its leaders, but many are under no obligation to stay, except for the desire to serve or financial incentive of cliff retirement. Desirable assignments can prove vital to career progression, family well-being, and job satisfaction. A transparent and fair allocation of jobs supports the Army mission.

Before work on an assignment slate can start, HRC must first identify Officers who will require a new assignment, either to support the Army Manning Guidance or due to the professional development requirements described in Army Regulation 600-3, *The Army Personnel Development System*. Assignment Managers Officers to develop a supply of Officers available for moves during the planned period or *assignment cycle*. Units currently under-filled due to attrition (retirements, separations, etc.) or projected moves of Officers then submit requisitions for replacement Officers. These requisitions are collated and prioritized based upon the strategy developed by the Army Manning Guidance. Validated requisitions serve as a starting point for assignment managers to develop their assignment slate; in general these are the assignments that they must fill.

Currently, assignment managers have few tools with which they can develop quality assignment slates. Instead, they toil over spreadsheets, manually weighing Officer preferences with Army requirements. Changes occurring late in the assignment process, such as a directed assignment, the rejection of an Officer from a Joint assignment, or the diagnosis of a serious illness or injury can require the assignment manager to either rework large portions of a slate, change assignment orders already issued, or accept a solution of lower quality than could otherwise be achieved. This research seeks to provide an algorithmic process for assignment managers to develop quality assignment slates that are robust to the vagaries of life in the presence of uncertainty.

1.2 Problem Statement

> *We have a sacred trust to take care of Soldiers in every component. Together we will make the Army stronger.*
>
> –MG Thomas C. Seamands, 5 June 2015
>
> Commander, United States Army Human Resources Command [3]

The Army Officer assignment process is critical to the success of the Army. The ideal assignment process balances the needs of the Army with the career goals and preferences of the Officer. The size of the problem for a typical assignment manager can easily preclude manual assignment processes. Automated processes generally involves some mathematical representation of the problem which is usually solved using a linear programming formulation.

Mathematical programs yield assignment strategies optimal with respect to some solution characteristics defined in an objective function. Unfortunately initial solutions may not be final solutions, as changes to assignment demands and the life situations of Officers potentially change the suggested solution.

Linear programming algorithms are chaotic; small changes to inputs, such as an assignment no longer available, the addition of new assignments, or the restriction or directed assignment of an Officer to a subset of assignments, may cause dramatic changes in the output [4]. Implementation of the new assignment strategy indicated may necessitate massive changes to assignment orders. A practical assignment process cannot tolerate dramatic changes and the resultant turmoil to Soldiers and their Families.

This work examines alternative approaches for the Army Officer assignment problem. The Gale-Shapley algorithm is examined with respect to its applicability to the assignment problem, the quality of solutions compared to linear programming,

and the robustness of the assignment methodology with respect to changes to the inclusion of additional assignment constraints.

1.3 Approach

First, we develop comparable formulations for an appropriate instance of the Assignment Problem, with parameters drawn from a representative case provided by the United States Army Human Resources Command (HRC). Linear Programming (LP) and Stable Marriage approaches are used to develop initial solutions to a boolean programming assignment model. This model should seek to minimize the total preferences, or maximize the social welfare, of both Officers and Assignments. The Gale-Shapley Stable Marriage algorithm provides a stable solution. These initial solutions are stored, and the underlying model randomly perturbed through the addition of one or more new constraints, grounded in realistic scenarios faced by Assignment Managers at HRC. New optimal solutions are found based using warm and cold start L as well as several Stable Marriage variants. The results from each result are compared with respect to overall solution quality as well as the number of changed decision variables from the initial to final solution. Various implementations of the Stable Marriage Algorithm are examined with respect to both solution quality and the number of changes, or robustness of the approach. New insight is gleaned into when each approach might be preferred.

1.4 Assumptions

> *All models are wrong but some are useful.*
>
> –George E. P. Box [5]

Individual preferences provided for each Officer represent their ordinal rankings of assignments to which they might feasibly be assigned. These preference lists should be feasibly complete, in that for each officer the rankings are complete over the subset of all assignments for which they are eligible. The errors present in the model data provided, are assumed to be transcription errors as discussed in Appendix A.

While the use of ordinal numbers as cost coefficients implicitly infers a linear preference relationship where another monotonic function might be more appropriate, the benefit of more accurate value functions might not be commensurate with the costs of eliciting those preferences. Preferences might be revealed using decision analysis techniques, such as allocating a fixed budget of preference-weighting points across a set of assignments, such as those show in Figure 1. These hypothetical preference function show some alternative distributions of 120 "points," shown on the y-axis, over 15 possible assignments, on the x-axis, while retaining the same preference order.

However, given the observed errors in a relatively simple ordinal ranking system, as documented in Appendix A, the expenditure of resources to ensure proper execution of more complex elicitation techniques will likely be high. Given that the Gale-Shapely algorithm provides results with ordinal inputs, we assume that such inputs are sufficient to represent the preference values of our officer population. Further analysis may illuminate the degree to which greater fidelity can improve an LP-derived *optimal* solution, as well as examine the costs associated with such improvements.

1.5 Summary

This research provides HRC Assignment Managers algorithmic processes by which high quality assignment slates could be rapidly generated while providing robust Officer-to-assignment slates that do not incur many changes when reacting to last minute perturbations (e.g. an assignment being canceled, an officer becoming un-

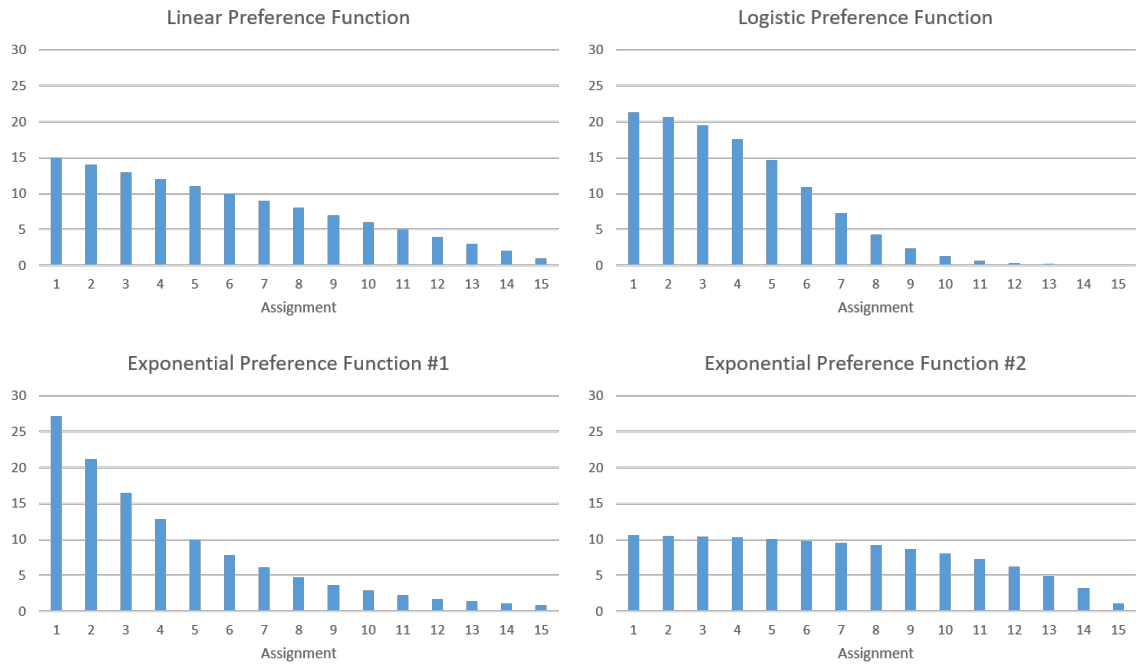


Figure 1. Theoretical Preference Functions of Officers over Potential Assignments

available for assignment). It may find further application to other matching problems in which there is a high cost or a long lead-time to implement changes to an incumbent solution. With higher penalties to changes in individual assignments, stability may be a desired characteristic in solutions that may not be incorporated via a single-dimensional fitness function. Such applications may include facility placement problems, the assignment of contracts, and the commitment of forces in combat models.

II. Literature Review

> We [previously managed] people in the Army basically by two variables: what is your rank and what is your occupational specialty... Now we have a million folks that we can tap into and get them on the field in the right position, in the right place at the right time.
>
> –LTG James McConville, 5 October, 2016
> Deputy Chief of Staff for Personnel
> Headquarters, United States Army [6]

2.1 Overview

This chapter reviews previously published literature on solution methodologies to the bipartite assignment problem. Solution approaches to the bipartite assignment problem can be roughly divided between those with a focus purely upon optimization and those approaches that utilize some form of the Gale-Shapley Stable Marriage Algorithm. We then explore previous applications of the Stable Marriage Algorithm, both in general as well as to specific military applications.

Of note, male and female pronouns noting the perspective of some arbitrary Army Officer are used throughout the discussion herein with neither preference nor prejudice towards any gender. Within this work and in context of the stable marriage algorithm, the terms *male* and *female* represent arbitrary labels established in applicable scientific literature for the two sets of a bipartite matching problem.

2.2 Bipartite Assignment Problem

Matching problems arise frequently. Ahuja [7] describes the classical bipartite matching problem as a special instance of the assignment problem wherein the objects are partitioned into two mutual exclusive groups. The two sets of interest are

sometimes called *men* and *women*, noted here as sets M and W respectively, each with complete preferences over the members of the other set such that the value or cost of any pairing can be evaluated. Bazaraa, Jarvis and Sherali develop a mathematical form of the assignment problem, expressed in terms of assigning m males (officers) to m females (assignments), shown in Equation (1) [8].

$$\begin{aligned} \text{minimize } & \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij} \\ \text{subject to} & \sum_{j=1}^{m} x_{ij} = 1, \qquad i = 1, ..., m \\ & \sum_{i=1}^{m} -x_{ij} = -1, \quad j = 1, ..., m \\ & x_{ij} \in \{0, 1\}, \qquad i, j = 1, ..., m \end{aligned} \tag{1}$$

Here c_{ij} is the cost or preference value associated with the *marriage* or pairing of man i to woman j and x_{ij} is a binary decision variable associated with the inclusion (i.e., $x_{ij} = 1$) or exclusion (i.e., $x_{ij} = 0$) of that *marriage* from a *matching*. Unequal cardinalities of sets M and W are typically handled by the inclusion of placeholder indices or dummy variables that represent a null partner with an appropriate null valued assignment contribution to the objective.

The minimization formulation would be appropriate for c_{ij} given in terms of costs. If benefits are instead used, a maximization formulation is more appropriate. However, it should be noted that these preference values are for each unique pair, and so they reflect some combination of male and female preferences. In other words, a *man* and a *woman* are allowed to have different preference values for each other; c_{ij}-values should reflect both preferences in that pair's contribution to the objective function.

A member of the solution is a pair formed from a single object from each group. Therefore, any solution S is a collection of mutually exclusive and collectively exhaustive pairs $(m, w) \in S$ such that $m \in M, w \in W$. A complete solution to the bipartite

matching problem is called a *matching*, which forms a bijection between the two sets M and W, herein labeled arbitrarily as men and women, respectively [7].

If the structure given in Equation (1) remains, we can simplify the expressions in vector-matrix form to that in Equation (2).

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) \\ &\text{subject to} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \qquad \text{where } \boldsymbol{b} = \begin{bmatrix} 1 \\ \\ \text{-1} \end{bmatrix} \\ & && \boldsymbol{x} \geq \boldsymbol{0} \end{aligned} \tag{2}$$

While the quality of a matching can be defined classically via a fitness function, such as the coefficients given in Equation (1), a more general scoring of a solution can be provided by some $f(\boldsymbol{x})$. The function f scores any solution, represented here by the vector of binary decision variables $\boldsymbol{x}$. Generally, f is assumed to be linear, which inherently assumes independent contributions of different matchings to the objective function.

Another approach to evaluating the relative quality of matchings is to use a list of ordinal preferences. Although Dean and Swar's work [9] does include the use ordinal preferences in allocation problems, not much other work found in the literature systematically addresses the comparative utility of ordinal value preference data to actual objective cost, preference, or value functions that are defined to have interval or ratio measures. Dean and Swar [9] introduced the *generalized stable allocation problem*, or GSAP, dealing with ordinal preferences, but they did not juxtapose the relative quality of solutions obtained through their proposed method to comparative instances of a *generalized assignment problem* with either ordinal cost components or representative costs. It appears to be an open question in the literature to establish, either empirically or theoretically, the marginal value of ratio preference values over

ordinal preference values.

Some literature ascribes value to other measures of matchings rather than just fitness derived from costs or preferences. Kimbrough and Kuo [10], in developing multiobjective fitness functions for matching heuristics, describe measures of *equity* and *social welfare*. These measures assign additional value to matchings that are somehow more "fair" to members of each gender. They define equity as the sum of absolute differences between the preferences of each match, and social welfare as the sum of ordinal preferences of each match [10].

To illustrate the difference between *social welfare* and *equity*, suppose there are two men, $\{m_1, m_2\}$ and two women, $\{w_1, w_2\}$ with both men and women uniformly preferring 1 to 2. We can see in Table 1 that while these two simple matchings have similar social welfare scores, their levels of equity differ. While the nomenclature connotes some meaning, care should be exercised to determine the applicability of these measures and their correct physical interpretation given some problem instance and application.

More commonly examined than *social welfare* and *equity* is the notion of *stability*. A matching S is deemed unstable if and only if $\exists(m, w) \notin S$ such that both m and w prefer each other to their respective partners within S [11]. Such a pair (m, w) is called a *blocking pair* [9]. A stable solution admits no blocking pairs. Although the literature on exact thresholds to define "nearly stable" solutions [12] is sparse, the stability of S is measured by the total number of blocking pairs it contains [10]. For any set of ordinal male and female preferences, there exists a stable matching that

Table 1. Comparison of Social Welfare, Equity and Stability

	Social Welfare	Equity	Stability
$\{(m_1, w_1), (m_2, w_2)\}$	$2 + 4 = 6$	$0 + 0 = 0$	0
$\{(m_1, w_2), (m_2, w_1)\}$	$3 + 3 = 6$	$1 + 1 = 2$	1

can be found in quadratic time [7]. In Table 1, we can see that, whereas the first match is stable, the second match is not.

2.3 Linear Programming Based Solutions to the Assignment Problem

The *minimum weighted matching problem on a bipartite graph* has some special properties that can be leveraged. Assignment problems, as special cases of Transportation problems, can yield bases that are unimodular [8]. In the matrix formulation of the assignment problem given in Equation (2), the matrix $\boldsymbol{A}$ may be totally unimodular, in that every square non-singular submatrix is unimodular. When A is totally unimodular and the right-hand-side is integer-valued, any basis will yield integer-valued decision variables. In other words, all extrema of the region $\{\boldsymbol{x}|\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$ have integer valued coordinates. Integrality constraints can then be relaxed when applying LP solution methods and an LP-based method will attain the optimal solution to the binary integer program, as long as any additional constraints retain this property of $\boldsymbol{A}$.

Unfortunately, the addition of other *side constraints* may lead to a matrix $\boldsymbol{A}$ that is not totally unimodular. In this case polynomial time algorithms may fail to provide integer-valued solutions to the LP relaxation. For instance, without transformation into an equivalent form, the Hungarian Algorithm cannot be applied and other polynomial time algorithms (e.g. Karmarkar's projective algorithm or the Affine Scaling algorithm) cannot solve the linear programming relaxation with a guarantee that the resultant optimal solution will be integral [8]. Other iterative methods of solving integer and boolean mathematical programs exist using cutting planes or branch-and-bound methods, and heuristic search methods forego a proof of optimality altogether. These methods can be used in concert and are typically integrated in commercial optimization software such as Gurobi [13] when solving combinatorial optimization

problems.

2.4 Stable Marriage Algorithm

The stable marriage algorithm, also called either the Gale-Shapley (GS) or Deferred Acceptance algorithm, is a heuristic that attempts to find a stable solution to the bipartite assignment problem, and is well studied in literature [14]. While it seeks to find a high quality solution, Gale and Shapley originally proposed it to address the college admissions process, a matching market with incomplete information, in order to reduce uncertainty for both colleges and applicants. In building the algorithm, Gale and Shapley reduced the problem to examining potential marriages between an equal number of men and women, each with ordinal or ranked preferences [15]. The bipartite assignment problem described in Section 2.2 is therefore often referred to as the *stable marriage problem*.

Within the context of the stable marriage approach to the assignment problem, the two sets of interest are commonly called *men* and *women*, noted here as M and W respectively, each with complete preferences over the members of the other set. As an assignment problem, a solution S is a collection of mutually exclusive and collectively exhaustive pairs (m, w) such that $m \in M, w \in W$. The goal of the Stable Marriage Algorithm, described in Algorithm 1, is to find a stable solution [15].

The Stable Marriage Algorithm requires as input two matrices each representing the ordinal ranking of members of one set on the other, i.e. men on women and *vice versa* [7]. These are represented in Algorithm 1 as $pref_m$ and $pref_w$ respectively. Alternatively, these may be viewed as a single matrix of ordered pairs [15], or as a set of preference vectors for each element [7], without loss of generality in either case. However stored, the preference information is used at each step in the algorithm.

At the first step of each iteration, the manner in which we select m from the set

Algorithm 1 Pseudocode for the Gale-Shapely Stable Marriage Algorithm

```
M is the set of Men, W the set of Women
for m ∈ M, w ∈ W:
pref_m, pref_w are preference lists for each men, women
S = ∅; the pairs (m,w) in the current matching

procedure STABLE MARRIAGE(pref_m, pref_w)
    while ∃ m ∈ M s.t.(m, w) ∉ S for some w ∈ W do
        m ← select m s.t. m ∉ p, ∀p ∈ S
        w ← pop(pref_m)
        p ← p ∈ S where w ∈ p
        if w ∉ p for some pair p ∈ S then
            S ← (m, w)
        else if w prefers m to current match then
            remove pair p from S
            S ← (m, w)
        else
            do nothing
```

of currently unmatched men may alter the final solution obtained, and we will later categorize the variants of Stable Marriage in part by whether we use a single arbitrary-but-consistent lexicographic order or we sample from the available men randomly. In any case, the GS algorithm will always return a stable solution [7].

McVitie and Wilson [16] developed a recursive algorithm that provides identical results to the GS algorithm, but the authors note that the larger consumption of memory would likely only result in moderate improvements in the algorithm's performance with respect to required computational effort. For test problems of size $m = 50$, they demonstrated a 30% improvement of the recursive algorithm over the GS algorithm [16]. However, these results with ALGOL, a functional programming language very supportive of recursion, are likely not achievable with Microsoft's Visual Basic for Applications (VBA), which trades-off features favorable for recursion for those more favorable to iterative methods [17].

McVitie and Wilson [16] also modify the recursive algorithm in order to enumerate

all stable solutions rather than just the single solution *optimal* to the proposing set (i.e., *male optimal*). An interesting note is that with the *female optimal* stable solution known, we can restrict our search since *men* can never be worse off than they are in the *female* optimal solution without constructing a *blocking pair* [16]. However, the $female\ optimal$ can be found directly by interchanging the roles of the proposals. Therefore, an implementation of this could be made to find first both the *male* and $female$ solutions before iteratively generating additional stable solutions.

When preferences contain some level of indifference, that is when one person has the same utility for a match with multiple partners, the algorithm can accommodate with an extended definition of *blocking pair*. Manlove [18] explored two variants of the Stable Marriage problem with ties (SMT). A weakly stable matching occurs when there are no blocking pairs that are strictly preferred, while a strongly stable matching admits no weakly preferred blocking pairs.

2.5 Previous Applications of the Stable Marriage Algorithm

Perhaps the best known application of the Stable Marriage Algorithm is its use by the National Resident Matching Program (NRMP). The NRMP implements the algorithm to provide stable matchings of medical residents to hospital residency programs and assurances to each program that their slots would be filled, without requiring medical students to respond in an unreasonably short time [19]. The success of this particular application was a driving force behind the awarding of the 2012 Nobel Prize in Economics to Roth and Shapely [20]. Similar uses of deferred acceptance algorithms are noted for several centralized labor markets in the medical community [21].

Given the original language of the Stable Marriage Algorithm in terms of college applications [15], it is unsurprising that its use in education has spread. Leveraging

the literature on matching markets, the Ministry of Education for Singapore utilizes the Stable Marriage Algorithm to match students to secondary schools using only student preferences and scores on the Primary School Leaving Examination [11]. The distribution of students among New York City high schools turned from lottery to a deferred acceptance algorithm in 2003 [22]. In 2005, Boston public schools adopted a deferred acceptance algorithm to assign students based upon parent preferences while balancing individual school requirements [23].

Applications of the Stable Marriage algorithm may also be found in modern information technology. A generalized version of the GS algorithm provides a rapid method to establish a matching between users and servers for content delivery networks, such as Akamai or Netflix. In content delivery networks, there is very rapid change of preference orderings and constraints due to time-varying network congestion as well as topological network changes over time. The algorithm provides a matching, stable over a short time horizon, of geographic clusters of users to network nodes that function as content servers [24].

The Stable Marriage Algorithm has begun to receive attention for its potential military applications as well. Hill et al. [4] applied the Stable Marriage Algorithm to the nuclear posture review, assigning a notional non-homogeneous supply of nuclear weapons to a list of targets. They observed chaotic behavior of LP-based solutions in dynamic-systems not seen in stable solutions, noting that changes may not be desirable when "adopting a plan already under execution" [4]. More recently, Naeem and Masood [25] of the Pakistani Military College of Signals explored the use of the Stable Marriage Algorithm to model detected threats and assign to them a probable target among some set of defended assets. This model enabled a larger threat evaluation and weapon assignment process by quickly providing a robust estimate of likely threat behavior [25].

In a human resources context, there have been cases of LP-based approaches in the services. In the United States Air Force, Wylie sought to optimize the assignment of so-called *rated officers*, those with aeronautical credentials. Wylie [26] utilized an assignment formulation that could be solved as a network flow with the routines available in SAS. Recently, Lepird [27] proposed further research into adapting algorithmic approaches to Air Force personnel assignments, using either an LP formulation or the Stable Marriage Algorithm. Within the United States Army, Sonmez and Switzer [28] used an agent-based model with an Officer-optimal stable solution, where side contracts could be chosen to receive a first choice. While the research primarily investigated the matching with contracts paradigm, the underlying model for their agents utilized a deferred acceptance algorithm.

III. Methodology

> Although the Army's industrial-aged personnel management system is adequate today, it will not support the Army's needs in 2025 and beyond. [We] recognize this and are calling for a human capital management transformation that will enable our effort to meet future strategic challenges more effectively.
>
> –Lieutenant General Robert B. Brown
> Commander, United States Army Combined Arms Center
> *Talent Management Concept of Operations for Force 2025 and Beyond* [29]

3.1 Introduction

Our methodology consists of comparing various implementations of LP and the GS algorithm both before and after an incumbent solution of each method is subjected to additional constraints and the same approach reapplied.

3.2 Provenance of the Exemplar Model Data

Given the common practice among Army assignment managers and the logistical burdens in eliciting more precise value functions from a large pool of personnel, the base models we build utilize readily available ordinal preferences for each Officer. We sourced both the preference data and officer data for these base models from anonymized exemplar data provided by HRC. Development of realistic assignment scenarios led to the additional constraints under review. While selection of the scenario was experimentally controlled, the selection of the affected Officers and assignments were randomized.

The data received from HRC required cleaning before use, yielding preferences for 161 Officers over 139 assignments. Some preferences were incomplete, or contained

errors in rankings. The steps taken to clean the data are listed in detail in Appendix A. Immediately, one Officer and assignment pair could be removed due to a unique training pipeline resulting in a single feasible assignment connecting an Officer's prior selection to enter that program. Dividing the available assignments into those considered "key and developmental" (KD), and dividing the Officers into those who require a KD assignment and those who do not, shows that since there are 86 non-KD Officers but only 71 KD assignments, we expected that every KD assignment should go to an Officer requiring a KD assignment.

3.3 Model Formulation

Sets.

Let the set of all Officers eligible for reassignment be $O \triangleq \{0, 1, 2, \ldots, 159\}$.
We partition O by whether the Officer has previously held a KD assignment for their current grade so that O_{KD} is the set of Officers lacking experience in a KD assignment, and O_B is the set of Officers who have already completed a KD assignment. By inspection we see that $O_{KD} \subseteq O$, $O_B \subseteq O$ and that $O_{KD} \cup O_B = O$ while $O_{KD} \cap O_B = \emptyset$, thereby establishing a partition of O.

O_B is the set of KD-Complete Officers $\triangleq \{0, 1, \ldots, 71\}$.

O_{KD} is the set of Officers requiring KD assignment $\triangleq \{72, 73, \ldots, 159\}$.

Similarly, we construct and partition the set of available assignments, $A_{available} \triangleq$

Table 2. Exemplar Data Provided by Sponsor

Type	Officers	Assignments
KD	86	71
Broadening	74	67
Special Training Program	1	1
Total	161	139

$\{0, 1, \ldots, 138\}$ into the first 72 KD assignments and the remaining 67 broadening (B) assignments, $A_{available} = A_{KD} \cup A_B$ respectively. In the current case, $A_{KD} = \{1, \ldots, 72\}$, and $A_B = \{73, \ldots, 138\}$ and $A_{KD}, \cap A_B = \emptyset$.

We also note two basic observations regarding the respective cardinality of each set, for this problem instance.

$$|O| > |A_{available}|. \tag{3}$$

There are more Officers than available assignments.

$$|O_{KD}| > |A_{KD}|. \tag{4}$$

There are more Officers requiring a KD assignment than there are KD assignments.

In response to the first observation, we augment the set of available assignments with a set of "dummy" or null assignments, A_D. With $A = A_{available} \cup A_D = A_{KD} \cup A_B \cup A_D$ such that $|A| = |O|$, we would interpret the assignment of an Officer to a element of A_D as the Officer not receiving any new orders and instead being deferred to a future assignment problem.

Officer preferences are recorded as 2-dimensional list, or alternatively a matrix, as shown in Equation (5). $\boldsymbol{P}$ is a matrix where p_{ij} is the preference value of the Officer i for assignment j.

$$p_{ij} \triangleq \begin{cases} \text{Preference over set } A_{KD}, & j \in A_{KD} \\ \text{Preference over set } A_B, & j \in A_B \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

There are two significant notes, however.

1. Due to problem structure and how preference information is collected, these preferences are split independently between preferences over A_{KD} and A_B.

2. In order to model indifference in LP, an average value is used. This is the average of the values that might have been given if a strict preference would have been required.

To illustrate the second point, we can examine the hypothetical preferences of two Officers over a group of assignments, identical except for a single ambivalence between two assignments. As shown in Table 3, we can see that Officer "B" is indifferent between his top two choices of assignment (Assignments 1 and 2).

Table 3. Problematic Indifference

Officer	Assignment 1	Assignment 2	...	Assignment n
A	1	2	...	n
B	1 (Indifferent)		...	$n-1$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	

Table 4 shows the pathological example of this behavior for a large number n assignments, with Officer B indifferent between all of the first $n-1$ assignments. A solution assigning the n^{th} assignment to Officer "B" is a 2^{nd} choice, while it would be the n^{th} for Officer "A". With a difference of $n-2$, we begin to see that the "rank" of the least preferred assignment might be reduced by the total quantity of indifference expressed by the Officer.

Table 4. Indifference - Pathological Example

Officer	Assignment 1	...	Assignment $n-1$	Assignment n
A	1	...	$n-1$	n
B	1 (Indifferent)			2

The cumulative effect of ties within a long list, if left unchecked, may leave Officers expressing preferences with ties at a disadvantage relative to their peers. While there

are several ways to handle ties, such as taking the maximum or minimum value of the tied values, as demonstrated in Table 5 we chose the average method without prejudice towards the others, in order to construct weakly stable matchings. In our context of assigning Officers to Assignments, the requirements of a strongly stable match are unnecessary, and the hypothetical issues raised by Manlove [18] do not apply.

Table 5. Indifference Expressed using the Average Method

Officer	Assignment 1	Assignment 2	...	Assignment 3
A	1	2	...	n
B	1.5 (Indifferent)		...	n

We define y_i as the year-group of Officer i. This is the cohort year that effectively marks the year of their entry into commissioned service, adjusted for prior experience or for unusual career advancement (e.g., promoted ahead or behind peers during previous promotion boards). Many Officer personnel management decisions in the Army, particularly consideration for promotions, are based upon an Officer's year-group. The parameter d_i is a relative measure of an Officer's year-group, such that

$$d_i = y_i - \min \boldsymbol{y} \tag{6}$$

and

$$d_{max} = \max_i \{d_i | i \in O\}. \tag{7}$$

We use the relative differences to determine for each assignment its preference over the set of Officers. We define c_{ij} as the coefficient expressing the preference of assignment j for Officer i.

$$c_{ij} \triangleq \begin{cases} \dfrac{s(d_{max} - d_i)}{10}, & j \in A_D, \\ sd_i, & i \in O_{KD} \text{ and } j \in A_{KD}, \\ s(d_{max} - d_i), & i \in O_B \text{ and } j \in A_B, \\ 0, & i \in O_B \text{ and } j \in A_B, \end{cases} \tag{8}$$

where s is a scaling constant equal to the size of the set O, or $s = |O|$. The scalar s is chosen so that the small maximum preference range of assignments is not dominated by the larger number of preferences expressed by Officers.

Decision Variables.

We define the decision to assign an Officer $i \in O$ to an assignment $j \in A$ as:

$$x_{ij} \triangleq \begin{cases} 1, & \text{if Officer } i \text{ is matched with assignment } j, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

We refer to the entire two-dimensional list, or matrix of decisions, as the matching given by $\boldsymbol{X}$. Since both matrices and sets of possible solutions are traditionally represented by capital letters, for clarity we denote the larger set of all possible matchings from which $\boldsymbol{X}$ is drawn as the universal set $\boldsymbol{U}$ such that $\boldsymbol{X} \in \boldsymbol{U}$.

Objectives.

We define several objectives for a matching of Officers to assignments. These functions map any match $\boldsymbol{X}$ to a real-valued measure. The first function measures match quality with respect to the utilization of KD assignments, the second function measures quality with respect to Officer preferences, and the third function measures the stability of a match by giving the number of blocking pairs. The fourth function

measures quality with respect to assignment preferences.

Objective 1: Maximize KD Opportunity.

$$f_1(\boldsymbol{X}) = \sum_{i \in O_{KD}} \sum_{j \in A_{KD}} y_i x_{ij}. \tag{10}$$

To optimize the available promotion opportunity of Officers, we prioritize the utilization of open key and developmental assignments to their fullest. A desired matching will assign non-KD complete Officers to KD assignments, further prioritizing those in the oldest year groups who are in closer proximity to their next promotion opportunity (i.e. those with the greatest need). Transformed for a minimization, Equation (10) uses the summed year-group values of all Officers requiring KD assignment that are matched with a KD assignment. Notice that if an Officer needing a KD assignment is matched to a broadening assignment, it contributes no value to the objective function. Likewise, the assignments of KD-complete Officers have no bearing on this value beyond the missed opportunity costs of occupying a KD assignment in lieu of an Officer who needs it.

Objective 2: Officer Preferences.

$$f_2(\boldsymbol{X}) \approx \boldsymbol{XP} = \sum_{i \in O} \sum_{j \in A} x_{ij} p_{ij}. \tag{11}$$

Equation 11 measures the quality of a matching by using the preference information of each Officer and the decision matrix $\boldsymbol{X}$. Minimizing this quantity seeks the greatest number of Officers their highest rated assignments, taking into account the varying preferences that Officers may have. As discussed both in Section 1.4, and in the definition of $\boldsymbol{P}$ above, this approximation is subject to error because the measurement

of ordinal preferences imposes a linear scale onto the values collected.

Objective 3: Stability.

$$f_3(\boldsymbol{X}) = \big\{\text{The number of blocking pairs contained in the matching } \boldsymbol{X}\big\}. \tag{12}$$

The third objective function measures the stability present in a solution. As noted in Section 2.2, a matching is stable if the number of blocking pairs is zero. Stability of a matching $\boldsymbol{X}$ is measured by the number of blocking pairs in that matching. The fewer the number of blocking pairs, the greater the stability of a match. To maximize stability we minimize the number of blocking pairs.

Objective 4: Assignment Preference.

$$f_4(\boldsymbol{X}) \approx \boldsymbol{CX} = \sum_{i \in O} \sum_{j \in A} c_{ij} x_{ij}. \tag{13}$$

This last objective function describes the overall assignment preference level for the matching, in a manner similar to the calculation of Officer preference in Objective 2.

3.4 Linear Programming Base Model.

The Linear Programming Model seeks to minimize the collective regret of Officers using an objective function similar to the measure of Social Welfare [10] discussed in Chapter 2. In this sense, we seek to minimize the regret of an Officer incurred by her assigned posting compared to that which would be personally optimal herself. Furthermore, we seek to simultaneously minimize the regret of assignments when

their assigned Officer is compared with the entire set. Of course, this optimization is subject to some additional constraints, representing the equities and requirements of the Army.

$$\begin{aligned}
\operatorname{lex}\min_{\boldsymbol{X}\in\boldsymbol{U}} \quad & \big(f_1(\boldsymbol{X}), f_2(\boldsymbol{X}) + f_4(\boldsymbol{X})\big) \\
\text{subject to} \quad & \sum_{i\in O} x_{ij} = 1, && \forall j \in A, \\
& \sum_{j\in A} x_{ij} = 1, && \forall i \in O, \\
& x_{ij} \in \{0,1\}, && \forall i \in O, j \in A,
\end{aligned} \tag{14}$$

By exploiting knowledge of the problem, specifically the structure of the set partitions and their influence on f_1, and using the approximation of f_2 in Equation (11) we can reformulate the multiobjective lexicographic minimization in Equation (14) into the optimization of Equation (15). We solve for the best attainable value of KD assignment utilization, k^*, and then add a constraint to Equation (15) using k^* as a constant.

$$\begin{aligned}
\min_{\boldsymbol{X}\in\boldsymbol{U}} \quad & \sum_{i\in O}\sum_{j\in A} x_{ij}(p_{ij} + c_{ij}) \\
\text{subject to} & \sum_{i\in O} x_{ij} = 1, && j \in A, \\
& \sum_{j\in A} x_{ij} = 1, && \forall i \in O, \\
& \sum_{i\in O_{KD}}\sum_{j\in A_{KD}} y_i x_{ij} = k^*, \\
& x_{ij} \in \{0,1\}, && \forall i \in O, j \in A.
\end{aligned} \tag{15}$$

$$
\begin{aligned}
\text{where } k^* = \min_{\boldsymbol{Z}\in\boldsymbol{U}} & \sum_{i\in O_{KD}} \sum_{j\in A_{KD}} y_i z_{ij} \\
\text{subject to } & \sum_{i\in O} z_{ij} = 1, && j \in A, \\
& \sum_{j\in A} z_{ij} = 1, && \forall i \in O, \\
& z_{ij} \in \{0,1\} && \forall i \in O, j \in A.
\end{aligned} \tag{16}
$$

The LP is modeled using the *numpy* [30] and *gurobipy* [31] packages of the Python programming language and solved using Gurobi's version 7.0.1 mixed integer programming solver [13]. Since the lower level program (16) is independent from the decision variables of the upper-level program in (15), k^* need only be found once and substituted into subsequent formulations as a parameter.

In solving the LPs above, we employ two methods. The initial solution for both LP variants is the same. The *cold start* LP does not "seed" the solver with these initial decision variable values, but rather will start from an initial feasible basis, such as a heuristically derived solution. The *warm start* LP uses the values of the decision variables in the initial solution as a guide, attempting to redress any infeasibility while searching for the new optimal value [31]. Since these methods attain an optimal solution we can state that, for identical instance, the two methods may diverge if and only if alternative optimal solutions exist. While alternative optimal solutions might exist in cases of a pair of Officers with identical preferences over some subset of assignments, additional solutions with the same objective value might also arise from more complex chain of "swaps" involving a larger number of Officers, but resulting in the same objective function value. We hypothesize that there should be no difference, on average, between the solutions obtained via these two methods.

3.5 Stable Marriage Model

As in the case of the LP, we extracted from the exemplar data the assignment preferences for each Officer. To properly represent Army priorities, we constructed a preference listing for each assignment over the set of available Officers. The resulting partially ordered set, or *poset*, of Officers will complement the preferences of the Officers over assignments. The GS deferred acceptance algorithm, or Stable Marriage Algorithm, is then used with these two posets to identify stable matchings.

In a traditional optimization formulation, as shown in Equation (17), we seek to first optimize the distribution of KD assignments, or f_1, and then generate a stable solution by minimizing f_3. The Stable Marriage Algorithm by design minimizes the number of blocking pairs by generating a stable solution, which by definition contains no blocking pairs. However, it is likely that there exist multiple matchings without blocking pairs, and the algorithm will only return one of multiple alternative optima.

$$\begin{aligned} \operatorname{lex}\min_{\boldsymbol{X}\in\boldsymbol{U}} \quad & \big(f_1(\boldsymbol{X}), f_3(\boldsymbol{X})\big) \\ \text{subject to} \quad & \sum_{i\in O} x_{ij} = 1, \qquad \forall j \in A, \\ & \sum_{j\in A} x_{ij} = 1, \qquad \forall i \in O, \\ & x_{ij} \in \{0,1\}, \qquad \forall i \in O, j \in A. \end{aligned} \tag{17}$$

The Stable Marriage Algorithm requires preference information for both assignments and Officers. While the preferences of Officer over assignments are available, we must construct the preferences for each assignment over the group of Officers such that the "Needs of the Army" represented by f_1 are taken into account. To do this, we construct a preference relation for each assignment based upon the nature of the assignment. While we could account for several factors, in this example our primary concern is the distribution of KD assignments.

For $i, j \in O$ and an assignment in A_{KD}, we establish a weak preference relation such that $i \succ j$ if either of the following conditions are true:

- Officer i requires KD and Officer j does not; $i \in O_{KD} \wedge j \in O_B$,
- Officer i has a lower or "older" year group than Officer j; $y_i < y_j$.

If both Officers have the same KD status and year group, then the assignment is indifferent between i and j, or $i \sim j$. Just as with ties among Officer preferences, there is a lack of a strict ordering (i.e. there are ties) within the preferences of the assignments.

We similarly construct a relation for Officers for assignments in A_B, preferring KD-complete Officers (O_B) to those requiring KD (O_{KD}) without prejudice to year group. While we model assignments in A_D with no preference, alternatively they could model some expressed preference with respect to how long or how often an Officer has previously been required to change duty locations, given by tour equity or date of their last move $\big($i.e., permanent change of station (PCS)$\big)$.

It should be noted that additional preference information for assignments, whether for certain experiences, skill identifiers, or manner of past performance could be easily incorporated at this point. Whether it is to ensure a nominative position is filled by a high performing officer or to match an assignment with a mechanized unit with an Officer with similar experience as a junior Officer, the options are limited only by the availability of data for use by assignment managers to implement a desired manning strategy.

The Stable Marriage Algorithm is then applied to determine a stable solution, with Officers proposing to assignments. Each Officer is then paired with the best assignment he might obtain without either another Officer being worse off. Assuming all Officers expressed their true preferences, then by the definition of stability

there should not exist any allowable "trade" among Officers that would improve the outcome for both Officers.

Since Kimbrough and Kuo [10]observed that solution characteristics might vary among stable matchings, we test several variants of the GS algorithm by varying how the proposing "male" or Officer is picked, whether the initial solution is used, and whether and how the best of multiple applications is used. The variants selected for test are shown in Table 6. For those that are described as *randomized*, the choice of the *male* or Officer at each step in the algorithm is by a random selection from the currently unmatched Officers. The lexicographic method simply chooses the next Officer in the list.

3.6 Perturbations to the Models

After the procedures in Sections 3.4 and 3.5 are applied, the resulting matching is stored as the incumbent solution and the respective model subjected to a combination of three possible alterations. Two additional possible changes are discussed and shown to be isomorphic to the three under study. For ten iterations at each combination, zero to five random instances of each additional constraint are generated. The incumbent solutions are then compared with the final solutions from the

Table 6. Variants of SMA and LP under test

Variant	Abbreviation
SMA, Randomized with Lowest Objective Value of 5 iterations	SMA-5o
SMA, Randomized with Fewest Changes of 5 iterations	SMA-5r
SMA, Randomized with Fewest Changes of 10 iterations	SMA-10r
SMA, Randomized with Fewest Changes of 30 iterations	SMA-30r
LP, Cold Start	LP-C
LP, Warm Start	LP-W
SMA, Lexicographic	SMA-Lex
SMA, Lexicographic with Warmstart	SMA-Lex/Warm
SMA, Randomized with Warmstart	SMA-Warm

updated model and the number of differences between the solutions are counted. In application, this approach of randomly selecting possible scenarios is known in the stochastic optimization field as a scenario optimization [32]. In addition to providing an immediate benefit of examining the Officer assignment decision under conditions of uncertainty, this approach also provides the opportunity to empirically compare the performance of the stable marriage heuristic with traditional linear programming techniques with respect to solution quality and stability.

Assignment Restriction.

An assignment restriction changes the model such that a subset of A becomes infeasible for that Officer. In practice, such an occurrence happens when life events cause an Officer to enroll in a program such as the Exceptional Family Member Program (EFMP) or the Married Army Couples Program (MACP). These programs may limit the assignment of an Officer, even to a desired posting. Another possibility might be duty limitations due to injury or illness preventing assignment to some units, such as Airborne status or those deploying in support of overseas operations.

Directed Assignment.

A directed assignment occurs when a particular Officer is directed to fill a particular assignment. Whether due to a by-name request (BNR) from a particular unit or the intervention of a senior decision maker into the process, these assignments take into account information that may not be routinely available to assignment managers.

Rejected Match.

Due to how Officers are assigned to Joint and Interservice assignments, also called Joint and nominative assignment, some units may exert influence over which Officer

they receive by effectively exercising a "veto" power. After an assignment manager "nominates" an Officer, a unit with such power undertakes its own review or interview process before approving or rejecting the nomination.

Unforecasted Assignment.

New assignment requirements sometimes occur. This might be due to rapidly unfolding situations, such as the establishment of a new joint task force headquarters (e.g., combat operations). Life events of an Officer in a high-priority unit, such as a new medical condition or legal action, might necessitate a reassignment. Depending upon priority and timing, the replacement might need to occur in the current assignment cycle. In application, this situation need not be explicitly modeled due to the presence of "dummy" assignments in the algorithms. Without any further preference information available for either Officers or assignments, there is no structural difference between an unforecasted assignment and the absence of an assignment. In practice, some degree of preference might be inferred, but the model examined herein assumes that no such information is available and does not explicitly model such information. However, should such information become available, it could be easily incorporated into the model.

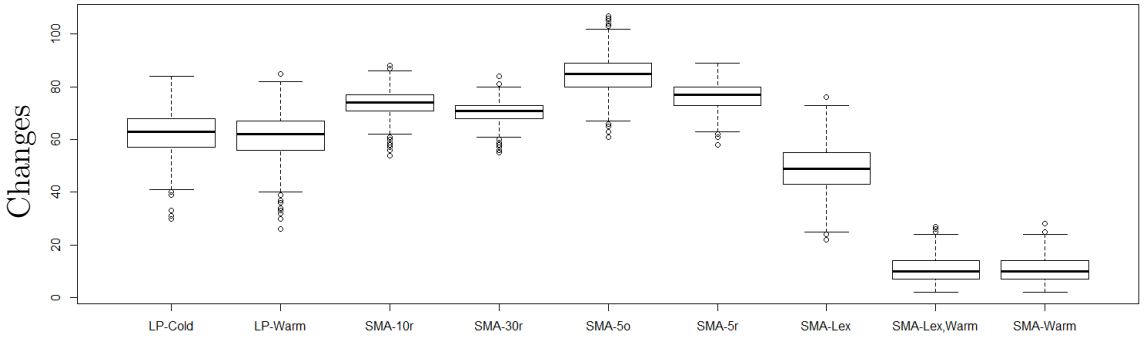
Removal of an Officer.

Life events, such as medical concerns, legal trouble, or personal career decisions might impact the availability of an Officer for reassignment. Despite many media portrayals, the Army's processes account for these in its reassignment decisions. Officers might retire in-lieu of a reassignment, or request a stabilization so that a high school-aged child does not have to move to a new school for their senior year. In application, this situation is analogous to a directed assignment described above, except

that the Officer is directed to a "dummy" assignment. As modeled, that Officer is effectively removed from the process.

IV. Analysis

4.1 Robustness of Approaches with Respect to Changes in Solution



Variant	Abbreviation
LP, Cold Start	LP-Cold
LP, Warm Start	LP-Warm
SMA, Randomized with Fewest Changes of 10 iterations	SMA-10r
SMA, Randomized with Fewest Changes of 30 iterations	SMA-30r
SMA, Randomized with Lowest Objective Value of 5 iterations	SMA-5o
SMA, Randomized with Fewest Changes of 5 iterations	SMA-5r
SMA, Lexicographic	SMA-Lex
SMA, Lexicographic with Warmstart	SMA-Lex,Warm
SMA, Randomized with Warmstart	SMA-Warm

Figure 2. Robustness (Changes to Incumbent Solution) By Method

Figure 2 shows the mean number of changes to an incumbent solution for a number of randomly chosen perturbations, where each of the three types are equally likely to occur. In this formulation, lower is better. The numeric values, along with 95% Scheffé confidence intervals [33], are shown in Table 7. All contrasts between the mean robustness of each methods are statistically significant, with the exception of the two LP methods. This indicates that for any two variants, except for the two LP methods, there is sufficient evidence to reject a null hypothesis that there is no difference between the mean robustness of the two approaches. For the warm and

cold start LPs, there is insufficient evidence to reject the assertion that there is no difference in mean robustness.

Table 7. Mean Number of Changes with 95% Scheffé Confidence Intervals [33]

Groups	Solution Method	95% Scheffé CI for Mean Changes
a	SMA, Randomized with Best Objective of 5 iterations	84.40 ± 6.94
b	SMA, Randomized with Fewest Changes of 5 iterations	76.52 ± 5.06
c	SMA, Randomized with Fewest Changes of 10 iterations	73.80 ± 4.73
d	SMA, Randomized with Fewest Changes of 30 iterations	70.51 ± 4.12
e	LP	62.31 ± 8.18
	LP with Warm Start	61.64 ± 8.24
f	SMA, Lexicographic	48.89 ± 8.76
g	SMA, Lexicographic with Warmstart	10.79 ± 5.12
g	SMA, Randomized with Warmstart	10.39 ± 4.76

4.2 Computational Effort

In Table 8, we examine computational effort, as measured by elapsed time, on an Intel Core i5-7200U CPU (2 Cores) with hyperthreading enabled running Ubuntu 16.10 in an Anaconda 4.2-managed Python 2.7.12 environment. Of note, most SMA iterations can be run in the same time as an LP and, given the context of the Army Assignment problem, there is no practical difference between even the smallest observed value (0.271 seconds) and the largest (7.39 seconds). The lack of practical difference, however, likely would not extend to larger scale problem instances and certainly is not generalizable to other problems.

4.3 Similarities of Cold Start and Warm Start Linear Programming

Further examining the previously noted similarity of the two LP methods, Table 9 highlights the further similarity of the two LP methods with respect to objective function value and the average choice of assignment given to Officers. With such

Table 8. Computational Effort as Elapsed Time with 95% Scheffé Confidence Intervals [33]

Groups	Solution Method	Mean Time(s)±95% Scheffé CI
a	SMA, Fewest Changes of 30 iterations	3.0069 ± 0.121
b	SMA, Fewest Changes of 10 iterations	1.1289 ± 0.034
c	SMA, Best Objective of 5 iterations	0.6592 ± 0.021
	SMA, Fewest Changes of 5 iterations	0.6592 ± 0.022
d	LP, Warm Start	0.6453 ± 0.020
	LP, Cold Start	0.6404 ± 0.022
e	SMA, Lexicographic	0.2811 ± 0.009
f	SMA, Randomized with Warmstart	0.2744 ± 0.006
	SMA, Lexicographic with Warmstart	0.2682 ± 0.014

high similarity in solution quality, computational effort, and number of changes, it is reasonable to conclude that there is no significant difference, on average, between the two variations of Linear Programming.

Table 9. Solution Quality of LP Methods, with 95% Scheffé Confidence Intervals [33]

Variant	Mean Objective	Average Officer Preference
LP Warm Start	70960 ± 265	16.03598 ± 0.769
LP Cold Start	70980 ± 301	16.08598 ± 0.784

4.4 Regression of Perturbation Type against the Number of Changes to Initial Solution

Relaxing the assumption that each type of perturbation under test occurs with equal frequency, we build a regression model to estimate the change attributable to each factor. The three factors are the number of restrictions, directed assignments and rejected assignments added after deriving the initial solution. We developed two models, one based upon the data observed for warm start LP and the other based upon the "best of five, changes focused" randomized SMA. The two regression models show that, while the two methods in general respond similarly to directed and rejected assignments, the algorithms respond much differently to the addition of assignment

restrictions. Lastly, the difference in intercepts reinforce observations noted in Figure 2, namely that for this dataset the LP method requires fewer changes on average than SMA.

LP Method.

We regress the warm start LP data against the observed changes to our initial solution. To remove non-constant variance in the response, we apply a Box-Cox transformation [34] to the number of changes, resulting in an estimated λ-parameter of 1.9. A model with all interaction terms was iteratively reduced in R [35], yielding a model with no interaction terms with all terms significant at the 0.001 level. A Breusch-Pagan [36] test failed to reject a null hypothesis of homoscedasticity. Variance inflation factors, approximately one, indicate there are no issues with multicollinearity of the data. While there are three potentially large (> 3) externally studentized residuals, in such a large data set three observations this extreme match those expected from a t-distribution [35].

$$\text{changes}_{\text{LP-Warm}} = (1000.03 + 83.63 n_{restrictions} + 81.52 n_{directed} + 58.70 n_{rejected})^{\frac{1}{1.9}} \quad (18)$$

Equation (18) shows the regression model, where $n_{restrictions}$ is the number of assignment restrictions added, $n_{directed}$ is the number of directed assignments and $n_{rejected}$ is the number of assignments rejected in the incumbent matching. For LPs, restrictions are the largest contributor to changes. LP is least robust to restrictions, and more robust to rejected matches.

SMA Method.

Fewest changes of five iterations with randomized selection (SMA-5r).

Similarly, we construct a regression model for the "best of five" SMA method where the best is selected by lowest number of changes rather than objective function value. Box-Cox [34] suggested a λ-value of 2, accounting for non-constant variance in the response. We developed a model with the three factors and no interaction. The model and all terms were significant at the 0.05 level or better. A Breusch-Pagan test [36] failed to reject the null hypothesis of homoscedasticity, and as expected there was no indication of multicollinearity within the factors. Also, as with the LP regression model, the number of large externalized residuals falls within the expected range [35].

$$\text{changes}_{\text{SMA-5r}} = \sqrt{2625.60 - 11.56n_{restrictions} + 82.63n_{directed} + 61.95n_{rejected}} \quad (19)$$

Equation (19) shows the regression model for the SMA algorithm, where $n_{restrictions}$ is the number of assignment restrictions added, $n_{directed}$ is the number of directed assignments, and $n_{rejected}$ is the number of assignments rejected in the incumbent matching. A larger intercept shows the generally lower performance of this algorithm, but the interesting observation is the reduction in changes with an increase in assignment restrictions. The model predicts a decrease in the number of changes, on average, for an increase in the number of additional restrictions.

Warm Start (SMA-Warm).

For the responses of the warm start SMA, the Box-Cox method [34] suggested a transform with a λ-value of 0.7, resulting in the relationship shown in Equation

(20). Residual analysis and application of the Breusch-Pagan test [36] highlighted no problems with model adequacy.

$$\text{changes}_{\text{SMA-Warm}} = (0.734 + 0.072n_{restrictions} + 1.22n_{directed} + 0.98n_{rejected})^{\frac{10}{7}} \quad (20)$$

The coefficient value for the number of restrictions ($n_{restrictions}$) was statistically insignificant ($p = 0.395$). This means that based upon the observed data, there is insufficient evidence to reject an assertion that restrictions do not cause changes to the model, or that the true coefficient value is zero. Intuitively, we would expect a solution to change only with the probability that the restriction impacts the assignment made within the initial solution. Therefore, we reason *a priori* that restrictions impact the number of changes, but the magnitude of the effect is small. However, it is possible that the true value may even be negative, confirming the observation made for a cold starting SMA.

An empirical justification for retaining the term $n_{restrictions}$ results from examination of the predicted residual error sum of squares, or PRESS, statistics for models both with and without the term for the number of restrictions [35]. In Table 10 we see that there is a small decline in the sum of squares of the PRESS residuals, indicating some small increase in predictive power in the expanded model. Given our *a priori* reasoning above, and the observed PRESS values, we determined that the model in Equation (20) is appropriate.

Table 10. Prediction Residual Error Sum of Squares (PRESS) for SMA-Warm Regression Model

Model	PRESS Statistic
Restrictions, Directed and Rejected Assignments	78,817.37
Directed and Rejected Assignments	78,849.71

4.5 Solution Quality

Table 11 displays the average objective function value, $f_2 + f_4$ for each of the solution methods. We note that the use of lexicographic selection in the SMA, which appears to provide much of the reduction in changes, imposes a cost with respect to solution quality.

Table 11. Solution Quality with Groupings from 95% Scheffé Confidence Intervals [33]

	Objective Value	
Treatment	Mean	95% CI Group*
SMA-Lex	72780	a
SMA-Lex,Warm	72460	b
SMA-5o	72400	bc
SMA-5c	72380	c
SMA-10c	72380	c
SMA-30c	72360	c
SMA-Warm	72240	d
LP-C	70980	e
LP-W	70960	e
*Treatments sharing a letter are statistically equivalent		

4.6 Pareto-Optimality of Solution Approaches

Based upon the observed data, a trade-off appears to exist between reductions in the number of changes and reductions in objective function value. Lexicographic SMA and warm start LP are both Pareto-optimal approaches, in this bi-objective comparison of quality and robustness. For the two objectives of quality and robustness, an approach is Pareto-Optimal if changing approaches to improve one objective necessitates a decrease with respect to the other objective [37].

To compare the LP-Warm, SMA-Lex and SMA-Warm results, we construct a conditioning plot of the data as shown in Figure 3. Here the data for each method are plotted side-by-side, with solution quality (i.e., the objective function value) on

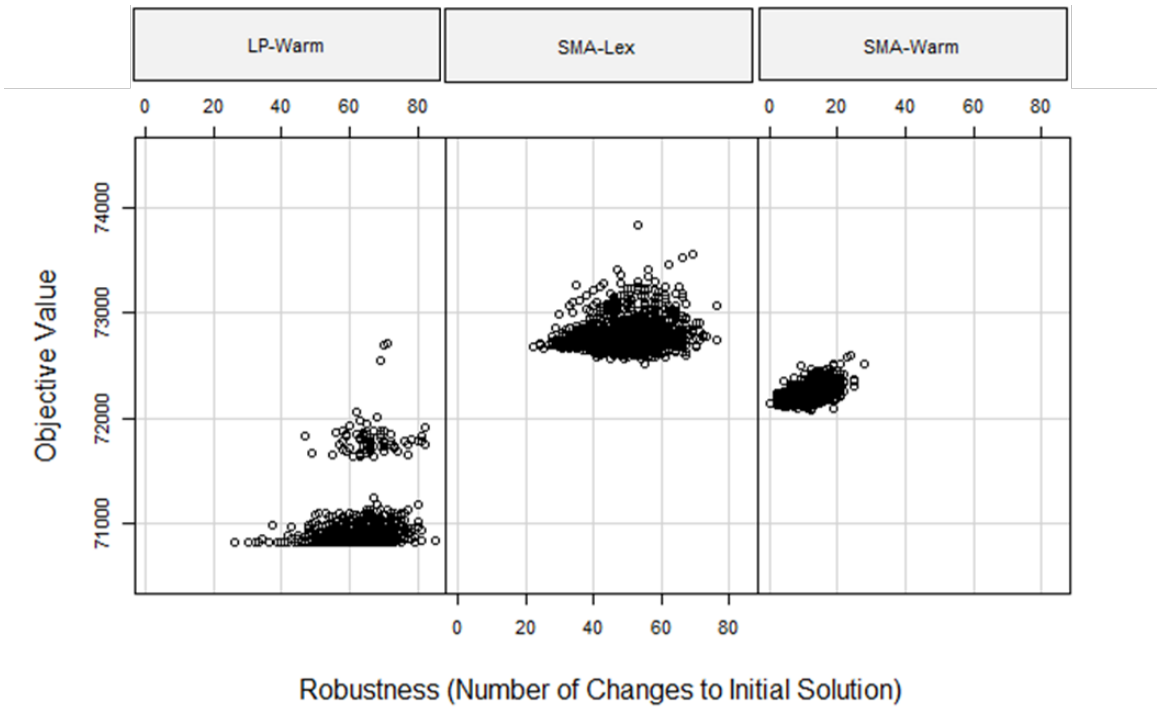


Figure 3. Plot of Efficient Frontier for Mean Solution Quality and Mean Robustness

the vertical axis and the number of changes observed on the horizontal axis. The lower left corner of each plot is the utopia point, optimizing both solution robustness and solution quality.

By inspection we can see that the LP method stochastically dominates the lexicographic SMA approach, in that we expect to perform at least as well if not better with respect to both measures when choosing LP over this SMA variant. With this observation, we can exclude this variant of SMA from the set of Pareto-Optimal approaches [37]. A comparison of the LP-Warm and SMA-Warm approaches shows that, in order to improve over the quality of SMA-Warm, we must change to an the LP-Warm approach that sacrifices robustness, and vice versa. Figure 4 simplifies the graph to examine only the means of each measure.

As both measures are formulated for minimization, a Utopian point is the closest point to the origin. On the graph, a notional Utopian point is noted for reference in the bottom left. We can clearly observe that the lexicographic variant (SMA-Lex)

lays on average behind the efficient frontier established by the two warm starting methods (SMA-Warm and LP-Warm). The relative weighting by a decision maker of the importance of quality over robustness will determine which side of the trade-off is preferable.

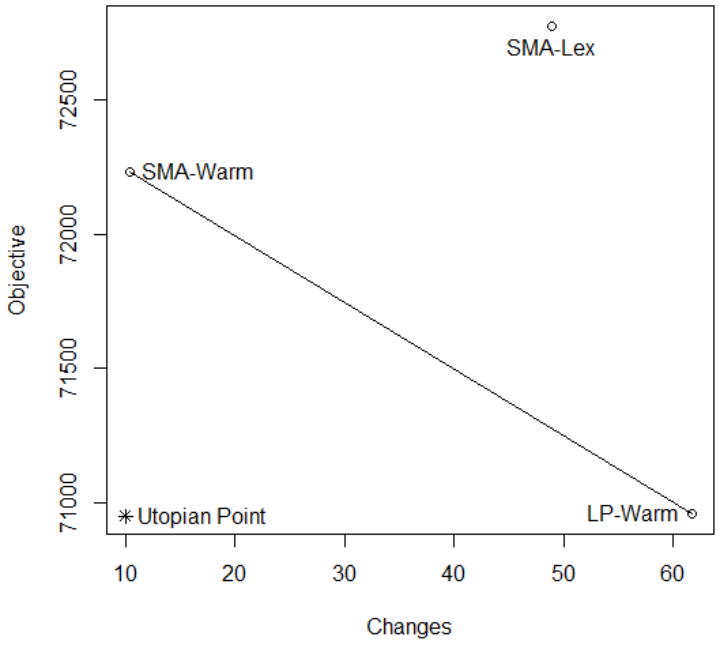


Figure 4. Plot of Solution Quality versus Robustness for Three Variants, with the Utopian Point Annotated

V. Conclusions and Future Research

> The Army's most important weapon is its people. Where the other services may man equipment, what we do is equip the Soldiers, the women and men who are the Army. That's where talent management comes into play.
>
> –LTG James McConville, 9 January 2017
> Deputy Chief of Staff for Personnel
> Headquarters, United States Army [38]

5.1 Conclusion

Pareto-Optimality of Approaches.

Based upon the observed data and our analysis of results in Section 4.6, we conclude for this problem instance that cold start SMA approaches do not appear Pareto-Optimal with respect to solution quality and robustness. More information is required from the decision-maker to examine the trade-off between the two measures, to ascertain is how much of a loss in solution quality should be sacrificed for solution robustness. If greater weight is given to robustness, then a warm-started SMA approach might increase robustness at the cost of quality. If instead solution quality is preferred, then a warm-started LP formulation is more appropriate.

Robustness Impacts of Differing Constraints on Approaches.

Assignment Restrictions.

The regression analysis performed in Section 4.4 appears to indicate that for SMA approaches robustness, as measured by the number of changes required to an incumbent matching after the inclusion of additional model constraints, appears to increase with an increase in the number of additional assignment restrictions. In other words,

increasing restrictions in a sparse preference structure appears to positively influence solution robustness in SMA approaches. This reaction can be contrasted with the observation that assignment restrictions are the most impactful constraint on LP formulations.

Directed and Rejected Assignments.

Directed assignments yielded the greatest impact over all SMA variants, whereas rejected assignments yielded the lowest impact on LP approaches. The impact of these observations depend upon an understanding of the relative frequency of these events occurring. Further observations or discussions with assignment managers may help inform follow-on analysis.

Development of an Officer-Assignment Matching Market.

Given the pervasive requirement to adjust the various methods used by Officers to communicate their preferences, discussed further in Appendix A, we note the benefit likely raised by automating the collection of Officer preference data through any standardized process with any form of integrated data validation. Such a process will likely reduce many transcription errors that might unnecessarily cloud the problem and reduce the quality of the resulting solution.

Development of a formal matching market system may ease the adoption of SMA by improving the availability of preference information elicited directly from Army leaders, without requiring an explicit mathematical modeling of a preference function for each assignment. The additional input leaders would have in determining the Officers that they receive might motivate an increase the preference information available to any algorithm used. If these preferences are collected as ordinal preferences with a low degree of indifference, then it may benefit from the application of

the GS algorithm.

5.2 Future Research

Will a Reduction in Assignment Indifference Result in an Improvement of Stochastic SMA approaches?

One characteristic of the problem under consideration is that the preference information for assignments over the set of all Officers is more sparse than the preference information elicited from the Officers. While this is partially an artifact of the available data, additional research could identify how a more granular preference structure might be reasonably developed. These might include the use of information regarding an Officer's manner of performance or the use of preference information elicited from units regarding Officer traits available in Army databases.

A new hypothesis from this observation is that, if preference information for the set of Assignments contained less indifference, the performance gap between LP and SMA approaches may be reduced. Follow-on work may test this hypothesis, the results of which can better inform when it might preferable to utilize any one methodology.

What potential impact does the use of ordinal preference structures have relative to the actual or theoretical preferences of Officers?

As noted in Chapter 2, there is scant literature regarding the impact of our study's assumption of a linear preference relationship of assignments by all Officers. Left to future investigation is an assessment of the full impact of this assumption, as well as a full accounting of the costs to gather the additional information. A comparison, either empirically or theoretically, of the marginal changes in solution quality between ordinal and ratio preference information can inform the potential benefits of increased information. This might be weighed against an accounting or more rigorous

estimation of the costs associated with eliciting a large number of preference functions. Once informed with a better understanding of the potential costs and benefits, Army leaders can make an informed decision regarding the continued use of ordinal information.

Relaxation of Pre-emptive Goal Program.

While the use of a lexicographic preference order in our LP formulation enabled its transformation into a relatively simple bi-level program, it is possible that a decision-maker's goals might not be so strict, allowing for some flexibility in the allocation of KD-Assignments. Developing a "Nearly Pre-emptive Goal Programming" approach would allow relatively minor deviations from the optimal value designated as k^* in our formulation. Given more flexibility, this modified LP approach might further improve the overall quality of the matching attained and better reflect the decision process of Assignment Managers.

Alternative Program Formulations.

A mathematical program might be formulated to minimize the number of deviations to an incumbent solution, or at least to incorporate and appropriately weight their number within the objective. Such a formulation may be able to provide additional approaches by which uncertainty could be addressed. A comparison to the methods described herein could prove enlightening, as a method to produce additional Pareto-optimal approaches to the problem.

Probabilistic Analysis of Scenario Frequency.

While this study examines the impact of the three scenario-based changes, it is likely that these scenarios do not occur with equal probability. The collection of data

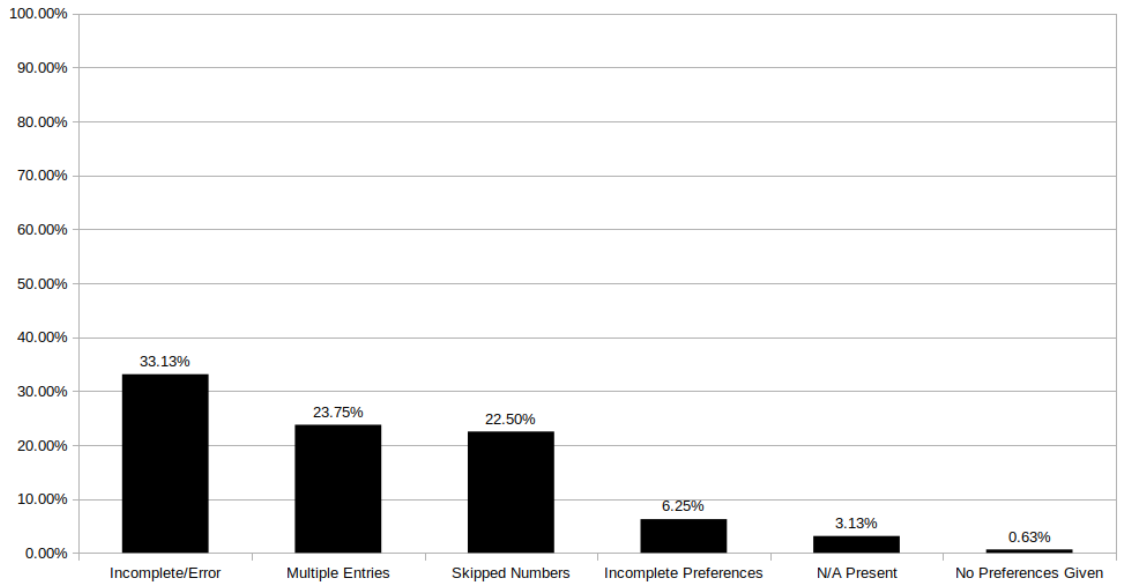
and additional study can inform how often each scenario occurs for the various populations, and whether it might change the approach recommended for some weighting of quality and robustness.

Appendix A. Data Cleaning

Reviewing the data received by the United States Army Human Resources Command (HRC), we discovered multiple instances of errors that might complicate any interpretation of Officer preferences. We summarize the various types of errors, how they are handled, and recommend an automated data validation schema appropriate for distributed and unsupervised data collection. We received an assignment worksheet with the collected ordinal preferences from a single assignment cycle for a single group of Officers.

We identified and characterized artifacts in the data as multiple preference values, skipped preference values, incomplete preferences, assignments annotated as "N/A" and a single Officer who did not express any preferences at all. Overall, about one in three Officer preferences contained at least one error or potential ambiguity, as noted in the first bar of Figure 5. The relative frequency of errors in the remaining bars does not sum to the overall error rate, as many preferences contained more than one type of error.

Figure 5. Observed Data Cleaning Errors



Most errors were likely typos, resulting from multiple iterations of copying and editing a list before submitting a final preference list to the assignment manager for manual transfer to the master copy. The coincidence of skipped numbers and multiple entries is intuitive. For instance, if there are two of one number and another number is later skipped, the final preference value will match the number of assignments, making it easier to overlook the compound error. While the majority of these artifacts were not proximate, we viewed that there was not an immediate concern and accept the values, as-is.

Multiple entries for a single preference value occur when an Officer assigns the same number to multiple assignments. We interpret this artifact as indifference between the two assignments and update our preference structure accordingly. While most of these errors might be rather innocuous, there is still the possibility for serious error. For example Officer 103 noted two assignments as her number 3 choice, but skipped the preference value 30. It is possible that, by assuming indifference between those assignments, we do not accurately model the true preferences of an Officer who possibly intended one to be assigned 30.

Incomplete preferences occur when no number at all was assigned to one or more assignments. We separately account for a single incident where not a single preferences was noted, but this is but an extreme case of incomplete preferences. Otherwise, the most extreme case was an Officer who submitted preferences over only 17 assignments. We also examined several cases where the Officer annotated "N/A" rather than an integer value. Upon further investigation, there was no information that indicated the Officer might be precluded from such an assignment. When no preference information is available, either by a blank or "N/A" entry, we assume that the Officer is indifferent between all such entries and append these to our preference structure accordingly.

Manual inspection and identification of these artifacts, as well as confirming any

interpretation via an individual contact with each Officer would likely be time and cost prohibitive. While an automated tool might help to identify such artifacts and even facilitate communication, the prevention of such errors would alleviate the problem at the source.

Appendix B. Code Documentation

assignmentFunction(initialSolution, allowableChanges, methodFlag): assignmentFunction returns a matching provided by a desired solution method (set by methodFlag) given a starting solution that is to be randomly altered according to the allowableChanges parameter. Solution time is also returned. assignmentFunction requires Python 2.7 with the ability to import the gurobipy, pandas and numpy packages (also inheriting integer division behavior from Python3).

Function inputs:

- initialSolution: a list of integer assignment indices such that initialSolution[i] = j iff Officer i is assigned to assignment j.
- allowableChanges: a list of integers representing the number of randomly chosen changes to the original assignment problem. For each change type i, j changes will be made iff allowableChanges[i]=j.

  allowableChanges[0]: *Assignment Restrictions.* Generate lists of assignments for allowableChanges[0] number of Officers such that the inclusion of any selected assignment for that Officer results in an infeasible matching. This change models widely impacting personal situations such as the Exceptional Family Member Program (EFMP), enrollment in the Married Army Couples Program (MACP), or deploy-ability restrictions such as medical profiles, or other restrictions on assignment with a wider impact on assignment feasibility.

  allowableChanges[1]: *Directed Assignments.* Randomly select allowableChanges[1] number of Officers and directs their assignment to a randomly selected assignment. This models the influence of a by-name request (BNR), or the

executive judgment of senior decision-makers acting upon additional information not routinely available to assignment managers.

allowableChanges[2]: *Rejected Match*. Randomly selects allowableChanges[2] number of Officers and determines if their current match (that is, the appropriate entry in initialSolution) is infeasible. This models the post-hoc influence of some individual assignments over the results of the Officer assignment process. Prototypically joint and nominative positions, these assignments can exercise a "veto" over the choice of a service's assignment manager.

– method flag: Designates the method by which the changed/new assignment problem instance is solved.

methodFlag = 0:*SMAWarmstart* Returns a matching found by using a Stable Marriage Algorithm, starting with the values in initialSolution.

methodFlag = 1:*SMAColdstart* Returns a matching found by using a Stable Marriage Algorithm, without specifying an initial solution.

methodFlag = 2:*LPWarmstart* Returns a matching found by using a Linear Programming model, with a warm-start using the values in initialSolution.

methodFlag = 3:*LPColdstart* Returns a matching found by using a Linear Programming model, without specifying an initial solution.

Function outputs: The function returns an object (of a "Solution" class or a tuple- depending upon implementation) consisting of:

– finalSolution: a list of integer assignment indices such that initialSolution[i] = j iff Officer i is assigned to assignment j.

– computationTime: a floating point representation of the computational effort expended on evaluation of the changed assignment problem. The exact determination of this value will depend upon implementations of timing routines in the underlying operating system.

Appendix C. Python Code

The following code was written for Python [39] using *numpy* [30] and *gurobipi* [13].

```
import numpy as np
import pandas as pd
import pickle as pkl

#two helper functions to save and load python objects
def save_obj(obj, name ):
    with open('data/'+ name + '.pkl', 'wb') as f:
        pkl.dump(obj, f, pkl.HIGHEST_PROTOCOL)
def load_obj(name ):
    with open('data/' + name + '.pkl', 'rb') as f:
        return pkl.load(f)

#Solution Class provides means of return more than one value
class Solution:
    def __init__(self, fS, rSF, c): #add msg?
          self.finalSolution = fS
          self.resultStatusFlag = rSF
          self.changes= c
          self.obj = 0.0
          self.feasible = 1

    def __str__(self): #when called via print
        return str([self.finalSolution, self.obj, self.feasible, self.changes])

    def __repr__(self): #when called interactively
        return str([self.finalSolution, self.obj, self.feasible, self.changes])
```

```
def setup():
    LPGetYearGroup()
    SMAGetPrefs()
    LPGetC()

#Generates 'p.pkl' file containing Officer preference
#dictionary only needs to be run once on a machine
#setup for use as a stack (pop/append) in reverse preference
#order. Stack contains sublists for each level of
#indifference (weak preference order)
def SMAGetPrefs():
    import numpy as np
    import pandas as pd
    p_kd = {}
    p_b = {}
    KD_Off= range(74,160)
    B_Off = range(0, 74)
    KD_Ass= range(0,71)
    B_Ass = range(73,139)
    D_Ass = range(139, 162)
    KD_D_Ass = B_Ass[0:15]

    Officers = list(set().union(KD_Off,B_Off))
    Assignments = list(set().union(KD_Ass, B_Ass, D_Ass))
    raw_pref = pd.read_csv('RawPreferences2.csv', dtype='str')
        #worst case, 72nd choice
        raw_pref.fillna(max(KD_Ass)+1,inplace=True)
```

```
for o in Officers:
    p_kd[o] = map(int,raw_pref.iloc[o,KD_Ass])
    for i in range(0,2):
        p_kd[o].append(999) #missing assignments 71/72
    for a in KD_D_Ass:
        p_kd[o].append(a)
    p_b[o] = map(int,raw_pref.iloc[o,B_Ass])
    for a in D_Ass:
        p_b[o].append(a)


save_obj(p_kd, 'p_kd')
save_obj(p_b, 'p_b')
Officers = list(set().union(KD_Off,B_Off))
Assignments = list(set().union(KD_Ass, B_Ass, D_Ass))
r_kd = {}
r_b = {}
raw_pref = pd.read_csv('RawPreferences2.csv', dtype='str')
for o in Officers:
    r=raw_pref.iloc[o,KD_Ass]
    r.fillna(max(KD_Ass)+1,inplace=True) #worst case scenario
    r_kd[o] = map(int,r)
r = raw_pref.iloc[o,B_Ass]
    r.fillna(max(B_Ass)+1,inplace=True)
    r_b[o] = map(int,r)
    for a in D_Ass:
        r_b[o].append(0)
save_obj(r_kd, 'r_kd')
save_obj(r_b, 'r_b')
```

```
#Generates 'y.pkl' file containing yeargroup list.
#Only needs to be run once on a machine
def LPGetYearGroup():
    d = pd.read_csv('OfficerData.csv')
    y = d['YG']
    y = y.values.tolist()
    save_obj(y, 'y')
    yg={}
    for o, year in enumerate(y):
        try:
            yg[year].append(o)
        except (KeyError, NameError):
            yg[year]=[o]
    save_obj(yg,'yg')


#Generates 'C.pkl' file containing cost coefficient list.
#Only needs to be run once on a machine
def LPGetC():
    KD_Off= range(74,160)
    B_Off = range(0, 74)
    KD_Ass= range(0,71)
    B_Ass = range(73,139)
    D_Ass = range(139, 162)
    y = load_obj('y')
    years = {}
    for i,year in enumerate(y):
        years[i] = (y[i]-min(y))
```

```
    Officers = list(set().union(KD_Off,B_Off))
    Assignments = list(set().union(KD_Ass, B_Ass, D_Ass))
    C = {}
    prefs ={}
    officer_preference = pd.read_csv('OfficerPrefs.csv')
    for o in Officers:
        prefs[o] = officer_preference[str(o)]
        for a in Assignments:
            if a in list(D_Ass):
                C[(o,a)] = 1.6 * (max(years)-years[o])
            elif not np.isnan(prefs[o][a]):
                if o in KD_Off and a in KD_Ass:
                    penalty = 160*years[o]
                elif o in KD_Off and a in B_Ass:
                    penalty = 160 *(max(years)-years[o])
                elif o in B_Off and a in KD_Ass:
                    penalty = 160
                elif o in B_Off and a in B_Ass:
                    penalty = 0
                else:
                    pass
                C[(o,a)] = prefs[o][a] + penalty
            else: #mildly infeasible, no KD prefs avail
                C[(o,a)]= 999
    save_obj(C,'C')


#Base SMA Code
def SMA(allowableChanges):
```

```
import random as rnd

sol = Solution([0,0], 1, 0)

#sets

KD_Off= range(74,160)

KD_Ass= range(0,71)

    B_Off = range(0, 74)

    B_Ass = range(73,139)

    D_Ass = range(139, 162)

    Officers = list(set().union(KD_Off,B_Off))

    #Assignments = list(set().union(KD_Ass, B_Ass))

    Assignments = list(set().union(KD_Ass, B_Ass, D_Ass))

    #Match KD Assignments

    KD_D_Ass = B_Ass[0:15]

    p_kd = load_obj('p_kd')

    p_b = load_obj('p_b')

    yg = load_obj('yg')

    y = load_obj('y')

    smaA = load_obj('smaA')

    KD_OtoA = {}

    BOtoA = {}

    res_list = {}

    #allowableChanges[0]: Assignment Restrictions

    avail0 = set(Officers)

    restrictions = rnd.sample(list(avail0), allowableChanges[0])

    for restriction in restrictions:

        res_list[restriction] = rnd.sample(Assignments, int(rnd.uniform

            (.05,.1) * len(Assignments)))

        for ass in res_list[restriction]:
```

```
        if ass in KD_Ass and restriction in KD_Off:
            p_kd[restriction][ass]  = max(Assignments) + 2
        elif ass in B_Ass:
            p_b[restriction][ass-min(B_Ass)] = max(Assignments) + 2
        else:
            pass
#allowableChanges[1]: Directed Assignment
taken = []
x = {}
avail0 -= set(restrictions)
availA = set().union(KD_Ass, B_Ass, D_Ass)
directeds = rnd.sample(list(avail0), allowableChanges[1])
for directed in directeds:
    #if directed in KD_Off:
        #KD_Off.remove(directed)
    #else:
        #B_Off.remove(directed)
    #x[directed]=rnd.sample(list(availA), 1)[0]
    if directed in KD_Off:
        x[directed]=rnd.sample(list(set(availA)-set(B_Ass)), 1)[0]
        KD_OtoA[directed] = x[directed]
    else:
        x[directed]=rnd.sample(list(set(availA)-set(KD_Ass)), 1)[0]
        BOtoA[directed] = x[directed]
    taken += [x[directed]]
    availA -= set(taken)
#allowableChanges[2]: Rejected Match
avail0 -= set(directeds)
```

```
rejects = []
while len(rejects) < allowableChanges[2]:
    reject = rnd.sample(list(avail0), 1)[0]
    if int(smaA[reject]) != 999:
        rejects.append(reject)
        avail0.remove(reject)
        if reject in KD_Off and smaA[reject] in KD_Ass:
            p_kd[reject][smaA[reject]] = max(Assignments)+2
        else: #if reject in KD_Off and smaA[reject] in B_Ass:
            #p_b[reject].remove(smaA[reject])
            p_b[reject][smaA[reject]-min(B_Ass)] = max(Assignments) + 2
del avail0
del availA
#Starting KD matching
opref = {}
apref = {}
unmatched = list(set(KD_Off)-set(directeds)) #listing of unmatched
    officers
noKD = [] #bookkeeping for KD Officers not getting a KD Assignment
#kdNotTaken = list(set().union(set(KD_Ass),set(KD_D_Ass) )- set(taken))
#for o in KD_Off:
level= {}
for assignment in KD_Ass:
    apref[assignment] = {}
    preference = 0
    for year in sorted(yg.keys()):
        preference += 1
        for officer in yg[year]:
```

```
            apref[assignment][officer] = preference
for assignment in KD_D_Ass:
    apref[assignment] = {}
    for officer in KD_Off:
        apref[assignment][officer] = -y[officer]
for officer in KD_Off:
    opref[officer] = p_kd[officer]
    level[officer] = 0
    if officer not in directeds:
        KD_OtoA[officer] = -1
while unmatched:
    officer = rnd.choice(unmatched)
        #officer = unmatched[0] #lexicographic
    if level[officer] > 100:
                noKD.append(officer)
                break
    try:
        possibility = opref[officer].index(level[officer])
        opref[officer][possibility]=-1
    except ValueError: #level not found in opref.index()
        level[officer] +=1
        continue

    if officer in directeds or possibility in list(taken):
        continue

    try:
        incumbent = KD_OtoA.keys()[KD_OtoA.values().index(possibility)]
```

```
        except ValueError:
            incumbent = -1

        if officer not in rejects or possibility != smaA[officer]:
                    possibilitypref = apref[possibility]
        else: #don't assign a reject to his incumbent match.
                    continue
        #no incumbent
        if incumbent == -1:
                    unmatched.remove(officer)
                    KD_OtoA[officer] = possibility
            if possibility in B_Ass:
                noKD.append(officer)

        #if assignment prefers officer to incumbent
        elif possibilitypref[officer] < possibilitypref[incumbent]:
                    KD_OtoA[incumbent] = -1
            if possibility in B_Ass:
                noKD.remove(incumbent)
                noKD.append(officer)
            unmatched.append(incumbent)
            KD_OtoA[officer] = possibility
            unmatched.remove(officer)
        else: #incumbent is preferred, do nothing
                    pass
    del p_kd        #garbage collect
    del yg
    #Setup for Broad. Matching
```

```
opref = {}
apref = {}
level = {}
B_OtoA ={}
p_b = load_obj('p_b')
unmatched = list(set(noKD + B_Off)-set(directeds))
for o in unmatched:
    opref[o] = p_b[o]
    if officer not in directeds:
        B_OtoA[o] = -1
        level[o] = 0
for assignment in B_Ass:
    apref[assignment]={}
    for officer in unmatched:
        if officer in B_Off:
            apref[assignment][officer] = 1
        else:
           apref[assignment][officer] = 2
for assignment in D_Ass:
    apref[assignment] = {}
    for officer in unmatched:
        apref[assignment][officer] = -y[officer]
while unmatched:
    officer = rnd.choice(unmatched)
    #officer = unmatched[0]     #lexicographic
    if level[officer] > max(Assignments)+2:
        print B_OtoA
        print str(level[officer]) +">"+str(max(Assignments)+2)
```

```
        raise ValueError
    try:
        possibility = opref[officer].index(level[officer])+min(B_Ass)
        opref[officer][possibility-min(B_Ass)]=-1
    except ValueError: #level not found in opref.index()
        level[officer] +=1
        continue
    try:
        incumbent = B_OtoA.keys()[B_OtoA.values().index(possibility)]
    except ValueError:
        incumbent = -1

    if officer not in rejects or possibility != smaA[officer]:
        possibilitypref = apref[possibility]
    else: #don't assign a reject to his incumbent match.
        continue

    if officer in directeds or possibility in list(taken):
        continue
    #no incumbent
    if incumbent == -1:
        unmatched.remove(officer)
        B_OtoA[officer] = possibility
        if possibility in KD_D_Ass:
            noKD.append(officer)

    #if assignment prefers officer to incumbent
    elif possibilitypref[officer] < possibilitypref[incumbent]:
```

```
            B_OtoA[incumbent] = -1
            if possibility in KD_D_Ass:
                noKD.remove(incumbent)
                noKD.append(officer)
            unmatched.append(incumbent)
            B_OtoA[officer] = possibility
            unmatched.remove(officer)

            #incumbent is preferred, do nothing
            else:
                pass
    for o in KD_Off:
        if o in noKD and o not in directeds:

            x[o] = B_OtoA[o]
        elif o not in directeds:
            x[o] = KD_OtoA[o]
    for o in B_Off:
        if o not in directeds:
            x[o] = B_OtoA[o]
    sol.finalSolution = []
    Officers = list(set().union(KD_Off,B_Off))
    for i in Officers:
        if x[i] in D_Ass or x[i] == -1:
            sol.finalSolution.append(int(999))
        else:
            sol.finalSolution.append(x[i])
    for restriction in restrictions:
```

```
            if x[restriction] in res_list[restriction]:
                sol.feasible = 0

        for reject in rejects:
            if x[reject] == smaA[reject]:
                sol.feasible = 0
        sol.resultStatusFlag = 0
        #save_obj(sol.finalSolution, 'smaA') #saved first time, used later

        sol.changes = [sum(i != j for i, j in zip(sol.finalSolution, smaA))][0]
    return sol
ef af(methodFlag=1, allowableChanges=[0,0,0]): #don't need init solution
    import numpy as np
    import gurobipy as gp
    import random as rnd
    sol = Solution([0,0], 1, 0)
    #Because Python doesn't have select—case
    if methodFlag == 0:   #SMA"Warmstart" —— may drop
        print "not implemented"
      elif methodFlag == 1: #SMAColdstart
                sol = bestOfN(30, allowableChanges) #Changed for various
                    methodologies
    elif methodFlag == 2: #LPWarmstart
                kd_m = gp.Model()
                y = load_obj('y')
                lpA = load_obj('lpA')

        #sets
```

```
KD_Off= range(74,160)

B_Off = range(0, 74)

KD_Ass= range(0,71)

B_Ass = range(73,139)

D_Ass = range(139, 162)

D_KD_Ass = range(139,154)

Officers = list(set().union(KD_Off,B_Off))

KD_Assignments = list(set().union(KD_Ass,D_KD_Ass))

Assignments = list(set().union(KD_Ass, B_Ass, D_Ass))

#handlemods/random perturbations

#allowableChanges[0]: Assignment Restrictions

restrictions = rnd.sample(Officers, allowableChanges[0])

A_r = {}

for restriction in restrictions:

A_r[restriction] = rnd.sample(Assignments, int(rnd.uniform(.05,.1) * len

    (Assignments)))


#allowableChanges[1]: Directed Assignment

A_d = {}

avail0 = set(Officers)

avail0 -= set(restrictions)

directeds = rnd.sample(list(avail0), allowableChanges[1])

availA = set(Assignments)

for directed in directeds:

        availA -= set(A_d.values())

        A_d[directed] = rnd.sample(list(availA), 1)

        A_d[directed]=A_d[directed][0]

#allowableChanges[2]: Rejected Match
```

```
availO -= set(directeds)
rejects = []
while len(rejects) < allowableChanges[2]:
        reject = rnd.sample(list(availO), 1)[0]
        if int(lpA[reject]) != 999:
        rejects.append(reject)
        availO.remove(reject)
#Phase I, slot KD Offs/objective  f_1
x={}
for o in KD_Off:
        for a in KD_Assignments:
                x[(o,a)] = kd_m.addVar(vtype=gp.GRB.BINARY, obj = y[o],
                    name="O{0:03d}".format(o) + "A{0:03d}".format(a))
kd_m.ModelSense = gp.GRB.MINIMIZE #MINIMIZE
for a in KD_Assignments:
        kd_m.addConstr(gp.quicksum(x[(o,a)] for o in KD_Off),gp.GRB.
            EQUAL,1)
        for o in KD_Off:
                if o in list(restrictions):
                if not list(set(A_r[o])&set(KD_Assignments)):
                        kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                            list(set(A_r[o])&set(KD_Ass))), gp.GRB.EQUAL
                            ,1)
                        kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                            KD_Assignments),gp.GRB.EQUAL,1)
                else:
                        kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                            KD_Assignments),gp.GRB.EQUAL,0)
```

```
        elif o in list(directeds):
        if A_d[o] in list(KD_Ass):
                        kd_m.addConstr(x[(o,A_d[o])], gp.GRB.
                            EQUAL,1)
                kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                    KD_Assignments),gp.GRB.EQUAL,1)
        else:
                kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                    KD_Assignments),gp.GRB.EQUAL,0)
        elif o in list(rejects):
        if int(lpA[o]) in list(KD_Ass):
                kd_m.addConstr(x[(o,int(lpA[o]))], gp.GRB.EQUAL
                    ,0)
                kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                    KD_Assignments),gp.GRB.EQUAL,1)
        else:
                kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                    KD_Assignments),gp.GRB.EQUAL,1)
        else:
                kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                    KD_Assignments),gp.GRB.EQUAL,1)


kd_m.update()
kd_m.setParam('OutputFlag', False)
#kd_m.write('lp.mps')
kd_m.optimize()
try:
        y_star = kd_m.objVal
```

```
except:
        return Solution(-1* np.ones(160), -1, -1)
del kd_m
C = load_obj('C')
x = {} #reallocating memory
bd_m = gp.Model()


for o in Officers:
        for a in Assignments:
                x[(o,a)] = bd_m.addVar(vtype=gp.GRB.BINARY, obj=
                    C[(o,a)],name="O{0:03d}".format(o) + "A{0:03
                    d}".format(a))


bd_m.ModelSense = gp.GRB.MINIMIZE #-1
for a in Assignments:
                bd_m.addConstr(gp.quicksum(x[(o,a)] for o in
                    Officers),gp.GRB.EQUAL,1)
for o in Officers:
        if o in list(restrictions):
        bd_m.addConstr(gp.quicksum(x[(o,a)] for a in A_r[o]), gp
            .GRB.EQUAL,1)
        bd_m.addConstr(gp.quicksum(x[(o,a)] for a in Assignments
            ),gp.GRB.EQUAL,1)
        elif o in list(directeds):
                bd_m.addConstr(x[(o,A_d[o])], gp.GRB.EQUAL,1)
        bd_m.addConstr(gp.quicksum(x[(o,a)] for a in Assignments
            ),gp.GRB.EQUAL,1)
        elif o in list(rejects):
```

```
                bd_m.addConstr(x[(o,int(lpA[o]))], gp.GRB.EQUAL,0)
                bd_m.addConstr(gp.quicksum(x[(o,a)] for a in Assignments
                    ),gp.GRB.EQUAL,1)
                else:
                bd_m.addConstr(gp.quicksum(x[(o,a)] for a in Assignments
                    ),gp.GRB.EQUAL,1)

#       bd_m.addConstr(gp.quicksum(gp.quicksum(y[o]*x[(o,a)] for a in
    KD_Ass) for o in Officers), gp.GRB.LESS_EQUAL, 1.0015* y_star)
        bd_m.addConstr(gp.quicksum(gp.quicksum(y[o]*x[(o,a)] for a in
            KD_Ass) for o in Officers), gp.GRB.LESS_EQUAL, y_star)
        bd_m.setParam('OutputFlag', False)
        bd_m.update()
        bd_m.optimize()
        sol.finalSolution = np.zeros(160)
                #try:
        for v in bd_m.getVars():
          if v.x >0:
              if int(v.varName[-3:])>=139 or int(v.varName[-3:]) in
                  [71,72]:
        sol.finalSolution[int(v.varName[1:4])] = 999
                 else:
                sol.finalSolution[int(v.varName[1:4])] = int(v.varName
                    [-3:])
        sol.finalSolution = list(sol.finalSolution.astype(int))
        sol.resultStatusFlag = 0 #Good Execution
        #save_obj(sol.finalSolution, 'lpA') #saved locally first time
            and referenced later
```

```
            sol.changes = sum(i != j for i, j in zip(sol.finalSolution, lpA)
                )
            sol.obj = bd_m.objVal
    elif methodFlag == 3: #LPColdstart
            kd_m = gp.Model()
            y = load_obj('y')
            lpA = load_obj('lpA')

            #sets
            KD_Off= range(74,160)
            B_Off = range(0, 74)
            KD_Ass= range(0,71)
            B_Ass = range(73,139)
            D_Ass = range(139, 162)
            D_KD_Ass = range(139,154)
            Officers = list(set().union(KD_Off,B_Off))
            KD_Assignments = list(set().union(KD_Ass,D_KD_Ass))
            Assignments = list(set().union(KD_Ass, B_Ass, D_Ass))
            #handlemods
            #allowableChanges[0]: Assignment Restrictions
            restrictions = rnd.sample(Officers, allowableChanges[0])
            A_r = {}
            for restriction in restrictions:
                    A_r[restriction] = rnd.sample(Assignments, int(rnd.
                        uniform(.05,.1) * len(Assignments)))

            #allowableChanges[1]: Directed Assignment
            A_d = {}
```

```
availO = set(Officers)
availO -= set(restrictions)
directeds = rnd.sample(list(availO), allowableChanges[1])
availA = set(Assignments)
for directed in directeds:
availA -= set(A_d.values())
A_d[directed] = rnd.sample(list(availA), 1)
A_d[directed]=A_d[directed][0]


#allowableChanges[2]: Rejected Match
availO -= set(directeds)
rejects = []
while len(rejects) < allowableChanges[2]:
reject = rnd.sample(list(availO), 1)[0]
if int(lpA[reject]) != 999:
rejects.append(reject)
availO.remove(reject)



#Phase I, slot KD Offs/objective  f_1
x={}
for o in KD_Off:
  for a in KD_Assignments:
        x[(o,a)] = kd_m.addVar(vtype=gp.GRB.BINARY, obj = y[o],
            name="O{0:03d}".format(o) + "A{0:03d}".format(a))
kd_m.ModelSense = gp.GRB.MINIMIZE #MINIMIZE


for a in KD_Assignments:
```

```
            kd_m.addConstr(gp.quicksum(x[(o,a)]
            for o in KD_Off),gp.GRB.EQUAL,1)
    for o in KD_Off:
            if o in list(restrictions):
            if not list(set(A_r[o])&set(KD_Assignments)):
                    kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                        list(set(A_r[o])&set(KD_Ass))), gp.GRB.EQUAL
                        ,1)
                    kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                        KD_Assignments),gp.GRB.EQUAL,1)
            else:
                    kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                        KD_Assignments),gp.GRB.EQUAL,0)
            elif o in list(directeds):
            if A_d[o] in list(KD_Assignments):
                        kd_m.addConstr(x[(o,A_d[o])], gp.GRB.
                            EQUAL,1)
                    kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                        KD_Assignments),gp.GRB.EQUAL,1)
            else:
                    kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                        KD_Assignments),gp.GRB.EQUAL,0)
            elif o in list(rejects):
            if int(lpA[o]) in list(KD_Ass):
                    kd_m.addConstr(x[(o,int(lpA[o]))], gp.GRB.EQUAL
                        ,0)
                    kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                        KD_Assignments),gp.GRB.EQUAL,1)
```

```
        else:
                kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                    KD_Assignments),gp.GRB.EQUAL,1)
        else:
                kd_m.addConstr(gp.quicksum(x[(o,a)] for a in
                    KD_Assignments),gp.GRB.EQUAL,1)


kd_m.update()
kd_m.setParam('OutputFlag', False)
#kd_m.write('lp.mps')
kd_m.optimize()
try:
        y_star = kd_m.objVal
except:
        return Solution(-1* np.ones(160), -1, -1)
#Continue with phase2, slot everyone- obj f_2
        C = load_obj('C')
x = {} #reallocating memory
bd_m = gp.Model()


        for o in Officers:
        for a in Assignments:
        x[(o,a)] = bd_m.addVar(vtype=gp.GRB.BINARY, obj=C[(o,a)
            ],name="O{0:03d}".format(o) + "A{0:03d}".format(a))


        bd_m.ModelSense = gp.GRB.MINIMIZE #-1
for a in Assignments:
                bd_m.addConstr(gp.quicksum(x[(o,a)]
```

```
        for o in Officers),gp.GRB.EQUAL,1)
for o in Officers:
        if o in list(restrictions):
        bd_m.addConstr(gp.quicksum(x[(o,a)] for a in A_r[o]), gp
            .GRB.EQUAL,1)
        bd_m.addConstr(gp.quicksum(x[(o,a)] for a in Assignments
            ),gp.GRB.EQUAL,1)
        elif o in list(directeds):
                bd_m.addConstr(x[(o,A_d[o])], gp.GRB.EQUAL,1)
        bd_m.addConstr(gp.quicksum(x[(o,a)] for a in Assignments
            ),gp.GRB.EQUAL,1)
        elif o in list(rejects):
        bd_m.addConstr(x[(o,int(lpA[o]))], gp.GRB.EQUAL,0)
        bd_m.addConstr(gp.quicksum(x[(o,a)] for a in Assignments
            ),gp.GRB.EQUAL,1)
        else:
        bd_m.addConstr(gp.quicksum(x[(o,a)] for a in Assignments
            ),gp.GRB.EQUAL,1)


bd_m.addConstr(gp.quicksum(gp.quicksum(y[o]*x[(o,a)] for a in
    KD_Ass) for o in Officers), gp.GRB.LESS_EQUAL, y_star)
bd_m.setParam('OutputFlag', False)
bd_m.update()
bd_m.optimize()


sol.finalSolution = np.zeros(160)
        try:
          for v in bd_m.getVars():
```

```
                        if v.x >0:
                          if int(v.varName[-3:])>=139 or int(v.varName
                              [-3:]) in [71,72]:
                    sol.finalSolution[int(v.varName[1:4])] = 999
                          else:
                            sol.finalSolution[int(v.varName[1:4])] = int
                                (v.varName[-3:])
                        sol.finalSolution = sol.finalSolution.astype(int
                            )
                        sol.resultStatusFlag = 0 #Good Execution
                        #save_obj(sol.finalSolution, 'lpA') #saved
                            locally first time and referenced later
                        sol.changes = sum(i != j for i, j in zip(sol.
                            finalSolution, lpA))
                        sol.obj = bd_m.objVal
                    except:
                        sol = Solution(-1* np.ones(160), -1, -1)

else: #Error
            print "Parameter Error: Invalid methodFlag"
            sol = Solution(-1* np.ones(160), -1, -1)

if methodFlag == 1:
        C = load_obj('C')
        sol.obj = 0
        for i,j in enumerate(sol.finalSolution):
            if j == 999:
                pass
```

```
                else:
                    try:
                        sol.obj = sol.obj + C[(i,j)]
                    except KeyError: #infeasible solution
                        print (i,j)
                        sol.obj = -1

    return sol


def eval(matching):
        C = load_obj('C')
        eval = 0
        for i,j in enumerate(matching):
            if j == 999:
                pass
            else:
                try:
                    eval = eval + C[(i,j)]
                except KeyError: #infeasible solution
                    sol.obj = -1
        return eval


def getRanks(matching):
        KD_Ass= range(0,71)
        B_Ass = range(73,139)
        r_kd = load_obj('r_kd')
        r_b  = load_obj('r_b')
        ranks = []
```

```
        for i,j in enumerate(matching):
            if j == 999:
                        pass
            else:
                        try:
                                if j in B_Ass:
                                        ranks.append(r_b[i][j-73])
                                elif j in KD_Ass:
                                        ranks.append(r_kd[i][j])
                                else:
                                        pass
                        except KeyError:
                                ranks = [0]
        if len(ranks) >0:
            return ranks
        else:
            return [0]



def bestOfN(N,aC):
        x={}
        bestObj = float("inf")
        bestPtr = 0
        for i in range(0,N):
                x[i] = SMA(aC)
                #if x[i].obj <bestObj and x[i].obj>0: #FOR BESTOF OBJ
                if x[i].changes <bestObj and x[i].changes>-1:
                    bestPtr = i
```

```
                bestObj = x[i].changes

    return x[bestPtr]
```

A Scenario-Based Parametric Analysis of Stable Marriage Approaches to the Army Officer Assignment Problem

MAJ Matthew Ferguson
Dr. Raymond Hill
Dr. Brian Lunday
Department of Operational Sciences (ENS)
Air Force Institute of Technology

RESEARCH OBJECTIVES

- **Examine the applicability of the Stable Marriage Algorithm (SMA) to the Army Officer assignment Problem**
- **Compare the quality of SMA solutions to those from an LP formulation.**
- **Evaluate the robustness of these assignment methodologies, in terms of the number of changes required after some random perturbation(s)**

MODEL FORMULATION

Sets

- KD-Complete, or Broadening, Officers $O_B \triangleq \{0, 1, \ldots, 71\}$.
- Officers requiring KD assignment $O_{KD} \triangleq \{72, 73, \ldots, 159\}$.

Similarly, we construct and partition the set of available assignments $A_{available} = A_{KD} \cup A_B$:

- $A_{KD} = \{1, \ldots, 72\}$
- $A_B = \{73, \ldots, 138\}$

Decision Variables

We define the decision to assign an Officer $i \in O$ to an assignment $j \in A$ as:

$$x_{ij} \triangleq \begin{cases} 1, & \text{if Officer } i \text{ is matched with assignment } j, \\ 0, & \text{otherwise.} \end{cases}$$

Algorithm Variants Under Test

Variant
SMA, Randomized with Lowest Objective Value of 5 iterations
SMA, Randomized with Fewest Changes of 5 iterations
SMA, Randomized with Fewest Changes of 10 iterations
SMA, Randomized with Fewest Changes of 30 iterations
LP, Cold Start
LP, Warm Start
SMA, Lexicographic
SMA, Lexicographic with Warmstart
SMA, Randomized with Warmstart

Objectives

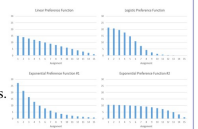
Objective 1: Maximize KD Opportunity.
$$f_1(\boldsymbol{X}) = \sum_{i \in O_{KD}} \sum_{j \in A_{KD}} y_i x_{ij}.$$
Objective 2: Officer Preferences.
$$f_2(\boldsymbol{X}) \approx \boldsymbol{XP} = \sum_{i \in O} \sum_{j \in A} x_{ij} p_{ij}.$$
Objective 3: Stability.
$$f_3(\boldsymbol{X}) = \{\#\text{ of blocking pairs in } \boldsymbol{X}\}.$$
Objective 4: Assignment Preference.
$$f_4(\boldsymbol{X}) \approx \boldsymbol{CX} = \sum_{i \in O} \sum_{j \in A} c_{ij} x_{ij}.$$

Random Perturbations (Additional Constraints)

For 10 iterations, choose zero to five random instances of the following scenarios:
Assignment Restriction: Officer is limited to some reduced set of Assignments, e.g. Exceptional Family Member Program (EFMP) enrollment
Directed Match: matches the selected Officer and assignment, e.g. removal of Officer or by-name request
Rejected Match: Cannot match Officer to current match, e.g. a "veto" for a nominative assignment

MAJOR ASSUMPTIONS

- This study assumes linear preference function, ordinal preferences can assume any number of possible relationships, some of which are shown to the right.



- This study also uses scenarios developed from realistic constraints that might occur later in the assignment process. While selection of the scenario was experimentally controlled, the selection of the affected Officers and assignments were randomized. The relative frequency of these events is yet unknown.

LP FORMULATION

$$\begin{aligned} \underset{\boldsymbol{X} \in \boldsymbol{U}}{\text{lex min}} \quad & (f_1(\boldsymbol{X}), f_2(\boldsymbol{X}) + f_4(\boldsymbol{X})) \\ \text{subject to} \quad & \sum_{i \in O} x_{ij} = 1, \ \forall j \in A, \\ & \sum_{j \in A} x_{ij} = 1, \ \forall i \in O, \\ & x_{ij} \in \{0, 1\}, \forall i \in O, j \in A \end{aligned}$$

LP REFORMULATION

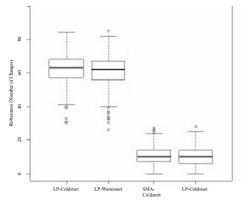
$$\begin{aligned} \min_{\boldsymbol{X} \in \boldsymbol{U}} \quad & \sum_{i \in O} \sum_{j \in A} x_{ij}(p_{ij} + c_{ij}) \\ \text{subject to} \quad & \sum_{i \in O} x_{ij} = 1, \quad j \in A, \\ & \sum_{j \in A} x_{ij} = 1, \quad \forall i \in O, \\ & \sum_{i \in O_{KD}} \sum_{j \in A_{KD}} y_i x_{ij} = k^* \\ & x_{ij} \in \{0, 1\}, \quad \forall i \in O, j \in A \\ \text{where } k^* = \min_{\boldsymbol{Z} \in \boldsymbol{U}} & \sum_{i \in O_{KD}} \sum_{j \in A_{KD}} y_i z_{ij} \\ \text{subject to} \quad & \sum_{i \in O} z_{ij} = 1, \ j \in A, \\ & \sum_{j \in A} z_{ij} = 1, \ \forall i \in O, \\ & z_{ii} \in \{0, 1\} \ \forall i \in O, j \in A \end{aligned}$$

SMA FORMULATION

$$\begin{aligned} \underset{\boldsymbol{X} \in \boldsymbol{U}}{\text{lex min}} \quad & (f_1(\boldsymbol{X}), f_3(\boldsymbol{X})) \\ \text{subject to} \quad & \sum_{i \in O} x_{ij} = 1, \ \forall j \in A, \\ & \sum_{j \in A} x_{ij} = 1, \ \forall i \in O, \\ & x_{ii} \in \{0, 1\}, \forall i \in O, j \in A \end{aligned}$$

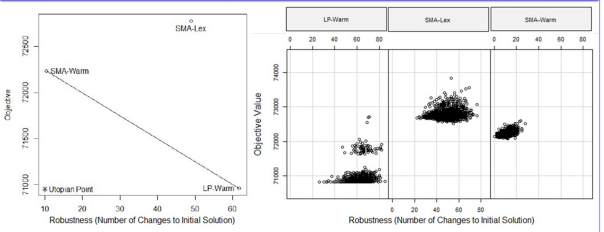
ROBUSTNESS: MEAN NUMBER OF CHANGES IN PERTURBED MODEL

Groups	Solution Method	95% Scheffé CI for Mean Changes
a	SMA, Randomized with Best Objective of 5 iterations	84.40 ± 6.94
b	SMA, Randomized with Fewest Changes of 5 iterations	76.52 ± 5.06
c	SMA, Randomized with Fewest Changes of 10 iterations	73.80 ± 4.73
d	SMA, Randomized with Fewest Changes of 30 iterations	70.51 ± 4.12
e	LP	62.31 ± 8.18
	LP with Warm Start	61.64 ± 8.24
f	SMA, Lexicographic	48.89 ± 8.76
g	SMA, Lexicographic with Warmstart	10.79 ± 5.12
g	SMA, Randomized with Warmstart	10.39 ± 4.76



Robustness (mean number of changes) for LP and SMA variants (above) and box-plot of robustness for LP and warm-start SMA variants (right)

PARETO-OPTIMALITY OF SMA AND LP WARMSTART APPROACHES



Comparing the robustness of approaches with the quality of solutions attained, we observe that a tradeoff condition exists:

- Warm-started SMA approach might increase robustness at the cost of quality.
- Warm-started LP formulation might increase quality with a loss in robustness

REGRESSION ANALYSIS OF ROBUSTNESS

Regression Models for Warmstart LP and SMA, and fewest changes for 5 randomized SMA iterations:

$$\text{changes}_{\text{LP-Warm}} = (1000.03 + 83.63 n_{restrictions} + 81.52 n_{directed} + 58.70 n_{rejected})^{\frac{1}{1.9}}$$
$$\text{changes}_{\text{SMA-Warm}} = (0.734 + 0.072 n_{restrictions} + 1.22 n_{directed} + 0.98 n_{rejected})^{\frac{10}{7}}$$
$$\text{changes}_{\text{SMA-5r}} = \sqrt{2625.60 - 11.56 n_{restrictions} + 82.63 n_{directed} + 61.95 n_{rejected}}$$

- For SMA approaches, inclusion of additional model constraints that restrict the feasible assignments for an Officer may actually improve robustness.

DEVELOPMENT OF A MATCHING MARKET

A large proportion of incomplete or missing data indicates a need for data validation. The establishment of markets could validate data, increasing data quality and speeding adoption by managers.

Sponsor: CPT Nick Paul, Army HRC

DEPARTMENT OF OPERATIONAL SCIENCES

Bibliography

1. "HRC mission and vision." https://www.hrc.army.mil/content/HRC%20Mission%20and%20Vision. Accessed: 2016-11-20.

2. "DOD personnel, workforce reports & publications." https://www.dmdc.osd.mil/appj/dwp/dwp_reports.jsp. Accessed: 2016-11-20.

3. R. Gall, "Seamands takes command of US Army Human Resources Command." https://www.dvidshub.net/news/165634/seamands-takes-command-us-army-human-resources-command. Accessed: 2017-01-09.

4. R. R. Hill, V. Chopra, B. Nadkarni, W. Cotsworth, and G. Schroeder, "Examining the stable marriage algorithm paradigm for agent-based optimization applications in support of defense planning challenges," in *IIE Annual Conference. Proceedings*, (Houston, TX), Institute of Industrial Engineers, May 2003.

5. G. E. Box, "Robustness in the strategy of scientific model building," *Robustness in statistics*, vol. 1, pp. 201–236, 1979.

6. C. T. Lopez, "New personnel system key to ferreting out untapped soldier talent." https://www.army.mil/article/176296/new_personnel_system_key_to_ferreting_out_untapped_soldier_talent. Accessed: 2017-01-09.

7. R. Ahuja, T. Magnanti, and J. Orlin, *Network flows: theory, algorithms, and applications*. Prentice Hall, 1993.

8. M. Bazaraa, J. Jarvis, and H. Sherali, *Linear Programming and Network Flows*. Wiley, 2010.

9. B. C. Dean and N. Swar, "The generalized stable allocation problem," in *International Workshop on Algorithms and Computation*, pp. 238–249, Springer, 2009.

10. S. O. Kimbrough and A. Kuo, "On heuristics for two-sided matching: Revisiting the stable marriage problem as a multiobjective problem," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pp. 1283–1290, ACM, 2010.

11. C.-P. Teo, J. Sethuraman, and W.-P. Tan, "Gale-shapley stable marriage problem revisited: Strategic issues and applications," *Management Science*, vol. 47, no. 9, pp. 1252–1267, 2001.

12. B. Aldershof and O. M. Carducci, "Stable marriage and genetic algorithms: a fertile union," *Journal of Heuristics*, vol. 5, no. 1, pp. 29–46, 1999.

13. "Gurobi optimizer features and benefits." http://www.gurobi.com/products/features-benefits. Accessed: 2016-10-29.

14. M. Fırat, C. Hurkens, and A. Laugier, "Stable multi-skill workforce assignments," *Annals of Operations Research*, vol. 213, no. 1, pp. 95–114, 2014.

15. D. Gale and L. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

16. D. G. McVitie and L. B. Wilson, "The stable marriage problem," *Communications of the ACM*, vol. 14, no. 7, pp. 486–490, 1971.

17. "Recursive procedures (visual basic)." `https://msdn.microsoft.com/en-us/library/81tad23s.aspx`. Accessed: 2016-10-15.

18. D. F. Manlove, "The structure of stable marriage with indifference," *Discrete Applied Mathematics*, vol. 122, no. 1, pp. 167–181, 2002.

19. A. E. Roth, "The evolution of the labor market for medical interns and residents: a case study in game theory," *The Journal of Political Economy*, pp. 991–1016, 1984.

20. "The Sveriges Riksbank prize in economic sciences in memory of Alfred Nobel." `http://www.nrmp.org/wp-content/uploads/2013/08/The-Sveriges-Riksbank-Prize-in-Economic-Sciences-in-Memory-of-Alfred-Nobel1.pdf`. Accessed: 2016-10-29.

21. A. E. Roth, "Deferred acceptance algorithms: history, theory, practice, and open questions," *International Journal of Game Theory*, vol. 36, no. 3, pp. 537–569, 2008.

22. "How game theory helped improve New York citys high school application process." `http://www.nytimes.com/2014/12/07/nyregion/how-game-theory-helped-improve-new-york-city-high-school-application-process.html?_r=0`. Accessed: 2016-10-29.

23. A. Abdulkadirolu, P. A. Pathak, A. E. Roth, and T. Snmez, "The Boston public school match," *The American Economic Review*, vol. 95, no. 2, pp. 368–371, 2005.

24. B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 3, pp. 52–66, 2015.

25. H. Naeem and A. Masood, "An optimal dynamic threat evaluation and weapon scheduling technique," *Knowledge-Based Systems*, vol. 23, no. 4, pp. 337–342, 2010.

26. A. M. Wylie, "Optimization of rated officer staff assignments," Master's thesis, Air Force Institute of Technology, 2007.

27. J. Lepird, "Position paper on a next generation air force personnel assignment system," tech. rep., United States Air Force, 2016.

28. T. Sonmez and T. B. Switzer, "Matching with (branch-of-choice) contracts at the United States Military Academy," *Econometrica*, vol. 81, no. 2, pp. 451–488, 2013.

29. U. S. A. C. A. Command, "Talent management concept of operations for force 2025 and beyond." http://usacac.army.mil/pubs/Force-2025-and-Beyond-Human-Dimension. Accessed: 2017-02-10.

30. S. van der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science Engineering*, vol. 13, pp. 22–30, March 2011.

31. "Gurobi optimizer example tour." `http://www.gurobi.com/documentation/7.0/examples.pdf`. Accessed: 2016-10-29.

32. R. S. Dembo, "Scenario optimization," *Annals of Operations Research*, vol. 30, no. 1, pp. 63–80, 1991.

33. F. deMendiburu, "agricolae tutorial." `http://tarwi.lamolina.edu.pe/~fmendiburu/`. Accessed: 2017-1-12.

34. W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*. New York: Springer, fourth ed., 2002. ISBN 0-387-95457-0.

35. R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

36. J. Fox and S. Weisberg, *An R Companion to Applied Regression*. Thousand Oaks CA: Sage, second ed., 2011.

37. M. Ehrgott, *Multicriteria optimization*. Springer Science & Business Media, 2006.

38. A. Dilanian and T. Akiwowo, "A new talent management program: An interview with lt. gen. james mcconville." `https://www.army.mil/article/179877/a_new_talent_management_program_an_interview_with_lt_gen_james_mcconville`. Accessed: 2017-01-09.

39. E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online; accessed 2016-10-22].

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
23-03-2017	Master's Thesis	Sep 2016 — Mar 2017

4. TITLE AND SUBTITLE

A SCENARIO-BASED PARAMETRIC ANALYSIS OF STABLE MARRIAGE APPROACHES TO THE ARMY OFFICER ASSIGNMENT PROBLEM

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Ferguson, Matthew D., MAJ, U.S. Army

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-MS-17-M-128

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

United States Army Human Resources Command
1600 Spearhead Division Ave
Fort Knox, KY 40121
COMM 502-613-6391
Email: nicholas.r.paul.mil@mail.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AHRC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This paper compares linear programming and stable marriage approaches to the assignment problem under conditions of uncertainty. Robust solutions should exhibit reduced variability in the presence of one or more additional constraints. Several variations of each approach are compared with respect to solution quality, as measured by the overall social welfare among Officers and Assignments, and robustness as measured by the number of changes after a number of randomized perturbations. We examine the contrasts between these methods in the context of assigning Army Officers among a set of identified assignments. Additional constraints are modeled after realistic scenarios faced by Army assignment managers, with parameters randomized. The Pareto efficient approaches, relative to these measures of quality and robustness, are identified and subjected to a regression analysis. The coefficients of these models provide insight into the impact the different scenarios under study, as well as inform any trade-off decisions between Pareto-optimal approaches.

15. SUBJECT TERMS

Optimization, Stable Marriage, Bipartite Matching, Assignment, Personnel, Human Resources, Army

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. R. Hill, AFIT/ENS
U	U	U	U	96	19b. TELEPHONE NUMBER *(include area code)* (937) 255-6565, x7469; Raymond.Hill@afit.edu