# A MULTI-OBJECTIVE APPROACH TO A BIPARTITE ASSIGNMENT MATCHING PROBLEM USING WEIGHTED VALUES FROM MULTIPLE CONSTRAINTS

THESIS

Greg S. Jeong, Captain, USAF

AFIT-OR-MS-ENS-10-05

**DEPARTMENT OF THE AIR FORCE**
**AIR UNIVERSITY**

**AIR FORCE INSTITUTE OF TECHNOLOGY**

**Wright-Patterson Air Force Base, Ohio**

AFIT-OR-MS-ENS-10-05

# A MULTI-OBJECTIVE APPROACH TO A BIPARTITE ASSIGNMENT MATCHING PROBLEM USING WEIGHTED VALUES FROM MULTIPLE CONSTRAINTS

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

Greg S. Jeong

Captain, USAF

March 2010

AFIT-OR-MS-ENS-10-05

A MULTI-OBJECTIVE APPROACH TO A BIPARTITE ASSIGNMENT MATCHING PROBLEM USING WEIGHTED VALUES FROM MULTIPLE CONSTRAINTS

Greg S. Jeong
Captain, USAF

Approved:

| | |
|---|---|
| \\SIGNED\\ | 15 March 2010 |
| Dr. Jeffery D. Weir (Co-Advisor) | date |
| \\SIGNED\\ | 15 March 2010 |
| Dr. Shane N. Hall, Maj, USAF (Co-Advisor) | date |

AFIT-OR-MS-ENS-10-05

## Abstract

US Air Force recruiters routinely assign new recruits to available jobs every month. The goal is to find the best assignments in an efficient manner. Although this problem is modeled as a bipartite assignment matching problem, it is not new to the field of Operations Research. This research presents a new approach to solve assignment matching problems given multiple side constraints. Using two multi-criteria optimization techniques, lexicographic optimization and the elastic constraint method, the assignment matching algorithm efficiently produces an optimal solution in a fraction of the time currently spent. This approach is demonstrated in assigning new USAF recruits to available jobs in any given month for the 369th Recruiting Squadron. The current 369th process manually creates assignments and can take weeks to complete, whereas the assignment matching algorithm takes less than two seconds and, on average, shows an increase in user defined goals of at least 15%.

*To my family and friends who have taught me to never give up in the face of adversity and to my father who taught me it is never too late to keep learning.*

## Acknowledgments

I would like to express my sincere appreciation to my committee. If it were not for their guidance, encouragement, and patience, I would have been lost throughout the course of this thesis. I would like to also thank the 369th Recruiting Squadron who provided a valuable research topic which could positively impact the future of the recruiting assignment process. I would also like to thank the AFIT faculty and staff for the opportunity to expand my knowledge to benefit not only my future, but the futures of those close to me and the US Air Force.

I am also deeply indebted to my classmates for their help and patience during my time at AFIT. The program would have been extremely difficult without their help, especially when it came to coding.

Thank you to my family and friends for their continued guidance and support throughout all my endeavors. I would not be the person that I am today if it were not for the love and support of my family and friends.

Greg S. Jeong

## Table of Contents

## List of Figures

## List of Tables

## List of Models

A MULTI-OBJECTIVE APPROACH TO A BIPARTITE ASSIGNMENT MATCHING PROBLEM USING WEIGHTED VALUES FROM MULTIPLE CONSTRAINTS

# I. Introduction

## Background

Military men and women are a significant resource within the United States, contributing to the country's strong global presence. A 2009 article in the *Air & Space Power Journal* stated that the Air Force is looking to enlist nearly 32,000 new recruits in 2009 not only to meet the Air Force's need for sustainment, but also to enlist recruits with the right skills at the right time (Marsman 2009). In order to maintain that image, recruiters diligently work to find quality recruits who have the potential to become future leaders.

The majority of potential future military leaders begin at the recruiting stage when they select a Service to pursue. Several qualities are considered when recruiters determine a recruit's eligibility for military service, including age, citizenship status, education level, use of drugs, and any physical and medical conditions that could prohibit enlistment. Once a recruit is considered eligible, administration of the ten Armed Services Vocational Aptitude Battery (ASVAB) tests determine enlistment eligibility and qualifications of potential recruits for military jobs. Four of the ten tests within the ASVAB make up the Armed Forces Qualification Test (AFQT), which measures the trainability of each potential recruit and predicts job performance. The AFQT is the main indicator of recruit aptitude (DoD 2002).

If a qualifying score is met and the recruit wishes to continue the process, the recruit undergoes a physical examination to determine physical fitness for military service and a background check to determine moral character standards at a Military Entrance Processing Station (MEPS). The physical examination usually consists of a standard medical screening to include measuring blood pressure, hearing, visual acuity, drug and human immunodeficiency virus (HIV) testing, as well as looking at prior medical history. If a medical problem is detected during the physical examination, the recruit may be required to get treatment before continuing with the recruitment process. Otherwise, if a disqualifying medical condition exists, the recruit may require a Service waiver prior to enlistment. To measure the moral character of potential recruits, a background check is performed to examine financial and criminal histories. If financial problems do exist, most recruits will not overcome financial hardships with junior enlisted pay. If a criminal history is present, a recruit's status is determined by the particular Service on a case-by-case basis depending on the nature of the criminal history (DoD 2002).

Once a recruit meets the respective qualifications, a meeting is arranged at MEPS with an occupational counselor, who discusses enlistment options such as possible jobs the recruit can fill, what the Service has to offer, and possible incentives for specific jobs. Usually, higher ASVAB test scores allow for more job choices. In some cases, the counselor may propose incentives to persuade recruits to choose positions that are difficult to fill (i.e., hard-to-fill jobs) such as Combat Control, Pararescue, or Linguist. For example, not every potential recruit is able to meet the physical rigor for a Pararescue position and not every potential recruit has the language aptitude to fill a Linguist position. Also, not every potential recruit would necessarily want to fill these types of positions, making it difficult to fill these types of jobs. Essentially, the last step of the recruitment process involves whether to enlist (and when) or to not enlist. Most recruits, upon accepting an enlistment offer, enter the delayed entry program (DEP), giving the recruit up to one year before the new recruit reports for duty (DoD 2002).

For US military organizations, there are hundreds of jobs new recruits can choose from such as special operations, communications, intelligence, and acquisitions. Depending on the requirements for each job and the objective(s) of the organization, a new recruit could potentially fill any available job. In May 2009, the Secretary of the Air Force Public Affairs stated that the Air Force will spend over $640 million in fiscal year 2010 for recruiting and retaining critical wartime skills such as explosive ordnance disposal (EOD), medical, intelligence, contracting, and special operations (Lyle 2009). Additionally, an article in *The Washington Post* (October 2009) stated that with the current state of the economy and the percentage of unemployment increasing, military recruiters have been able to meet their objectives in both numbers and quality recruits for all components of active duty and reserve forces (Tyson 2009).

Increased availability of quality recruits inherently suggests that available jobs can be filled by those recruits. For the Air Force and sister Services, filling available jobs with those quality recruits begins with recruitment. The quality of a recruit is comprised of several factors including physical fitness and skill levels, medical conditions, and/or ASVAB test scores. Once assessed, new recruits are assigned to available jobs using the personnel assignment system. Specific objectives are considered when assigning a new recruit to an available job, including fulfilling requirements for ethnic representation within a job category and/or maximizing the number of assignments of quality recruits to available jobs. A number of constraints must also be considered during the recruitment process, such as the number of new recruits and vacant jobs that are available in a given timeframe, the recruit’s job preference, and the recruit’s quality.

To illustrate this problem, suppose there are 100 recruits of the same ethnic background with high AFQT scores and that all recruits prefer any job. In that month, there are 100 total jobs available. Also, suppose that each recruit can meet the requirements for each available job in that month. For this scenario, there are 100! ways the recruits can fill jobs (i.e., 100 ways to fill the first job, 99 ways to fill the second job, 98 ways to fill the third job, etc.), but the recruiter can easily assign recruits to jobs within a matter of minutes by sorting the recruit list by the latest the recruit is willing to wait for a job and use the revised list to assign recruits to jobs in descending order. Now suppose not all recruits have high AFQT scores, but each recruit can still meet the requirements for each available job in that month. Again, the recruiter can easily assign recruits to jobs within a matter of minutes by sorting the recruit list by highest AFQT score first and then sort by the latest the recruit is willing to wait for a job and use the revised list to assign recruits to jobs in descending order. For a more complicated scenario, suppose the recruits' job preferences are a factor, the recruits belong to different Classes which may affect the recruits' quality (depending on the organizational objectives), and not every recruit can meet the requirements for each available job in that month. With multiple objectives (sometimes conflicting) and various constraints to consider, trying to manually accomplish this task to find an optimal set of assignments that will meet the user's objectives while balancing trade-offs and maintaining feasibility is a daunting task. This is where using optimization techniques to formulate a multi-objective weighted bipartite assignment matching algorithm helps to reduce the number of man-hours spent trying to manually solve instances of this problem.

## Research objectives, assumptions, and questions

The motivation for this research comes from the multi-objective problem which involves assigning new USAF recruits from the 369th Recruiting Squadron to available jobs in any given month. Creating a new method to find 'optimal' solutions (or good assignments) is critical for the 369th. Currently, assigning new recruits to available jobs is performed manually, taking up to two weeks to complete. The number of man-hours it takes to solve instances of this problem each month can be better utilized. Therefore, the primary goals of this research are to develop and validate a multi-objective weighted bipartite assignment matching algorithm that significantly decreases the number of man-hours spent to create a user-specified set of optimal assignments (i.e., goals/targets) and maximizes the number of assignments with highly 'valued' (or quality) recruits.

To develop the matching algorithm, model implementation using Visual Basic for Applications (VBA) with an optimization software package (LINGO) will require certain inputs to attain the desired outputs. With a specified weighting for each Class, the VBA matching algorithm keeps track of all recruits and jobs in a given month along with the recruit's availability to fill a particular job and determines the recruit's 'value'. For the 369th problem, a valued recruit is equal to the weighted scoring of preferred jobs and three Classes: ethnicity (i.e., African American and/or Hispanic) and an AFQT score $\geq 50$ (or CAT I-IIIA). Next, the VBA matching algorithm formulates the newly acquired information into a readable format for LINGO as input. With the resulting LINGO output, the VBA matching algorithm then translates the output into a readable format for the user that includes the specific recruit/job assignments, the percentage of assignments for each Class compared to the total number of assignments, the percentage of assignments for each Class compared to the maximum allowable assignments for each Class, the total number of assignments made, the maximum number of assignments each Class could achieve, the total number of assignments the matching algorithm finds for each Class given the weighting levels, and the total number of jobs with the percentage of jobs assigned.

Specifically, given a set of recruits with certain values and available jobs each month, the matching algorithm creates a set of feasible assignments that preemptively assigns all hard-to-fill jobs and then tries to assign all remaining jobs to recruits that are of certain ethnicity and skill level (i.e., an AFQT score $\geq 50$).

To verify and validate the matching algorithm, small randomly generated instances are created and past results are used for comparison. The small instances are used to compare the matching algorithm output against manual implementation. These instances are varied between: the total number of recruits and jobs (both $\leq 10$), recruits of different Classes, recruit and job availability, and job types (i.e., hard-to-fill jobs vs. non hard-to-fill jobs). When successful results are reached with the small instances, previous 369th instances are used to compare the matching algorithm output against past implementation. For the matching algorithm to be valuable, it must be able to reproduce past job assignments that are equal to or better in terms of user requirement(s) and number of assignments. It also must significantly decrease the number of man-hours spent to optimally assign new recruits to available jobs. However, the matching algorithm is still valuable if it takes the same amount of time (or longer) to assign new recruits to available jobs and produces better results compared to the current process.

This research assumes four key assumptions with respect to a given problem instance:

1) The list of job preferences each recruit requests indicate that the recruit is qualified for the job so there is no need for verification.
2) If the matching algorithm assigns a recruit to a job, it will be considered an assignment (i.e., the recruit will not deviate from that assignment).
3) A recruit may belong to more than one Class (e.g., a recruit may be considered African American and Hispanic with an AFQT score $\geq 50$).

4) If a recruit is assigned to a hard-to-fill job, there will be no trade-offs between hard-to-fill and non hard-to-fill jobs unless the trade-off results in assigning more hard-to-fill jobs.

There are several questions that are answered during the course of this research:

1) How much time will be saved using the matching algorithm this research provides versus the 369th Recruiting Squadron's current method?
2) Will this matching algorithm meet the user's defined percentage goals/targets set for each Class?
3) Will finding the maximum number of assignments versus maximum value of assignments (i.e., quantity versus quality of assignments) provide better results in terms of percentage of assignments for each Class?
4) Does this matching algorithm apply to other scenarios other than assigning new Air Force recruits to available jobs?

## Organization

This thesis is organized as follows. Section II describes the similarities and differences of past formulations and models to solve the assignment matching problem. This section is not collectively exhaustive, but the past formulations and models are critical to this research. Section III describes the methodology to solve assignment matching problem instances with the verification of the matching algorithm. Section IV describes the application with the results, analysis, and validation of applying the methodology to randomly generated and previous 369th instances. Section V states concluding remarks and possible future work. Note that for the remainder of this research, quality and value have the same meaning.

# II. Literature Review

This section explains the various works of previous research that use variations of the assignment matching problem. First is a brief background of the assignment matching problem, followed by previous research. Although the previous research used are not collectively exhaustive, they enable this research to focus on finding new ways to obtain optimal solutions for the assignment matching problem. Finally, the research contributions are discussed.

## Background

The assignment matching problem is a well known combinatorial optimization problem in the field of Operations Research (OR). There is a plethora of archived research for the assignment matching problem such as the generalized assignment problem, traffic assignment problem, quadratic assignment problem, and the job assignment problem (Kleeman and Lamont 2007). Many have used different types of assignment matching problems to solve various problem instances. But in order to understand the assignment matching problem, it is important to know where its origins are derived and how it has evolved before discussing the ways others have used this method.

The assignment matching problem is generally described as assigning a number of elements (e.g., people or machines) to a number of positions (e.g., jobs or tasks) with the goal of assigning all elements to positions given a certain cost element (Kleeman and Lamont 2007). Its roots come from the transportation problem which is as follows (Gass 1985):

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} x_{i,j} \qquad (1)$$

$$s.t. \ \sum_{j=1}^{n} x_{i,j} = a_i \qquad i = 1, 2, ..., m \qquad (2)$$

$$\sum_{i=1}^{m} x_{i,j} = b_j \qquad j = 1, 2, ..., n \qquad (3)$$

$$x_{i,j} \geq 0 \qquad \forall \ i, j.$$

Model 1. Transportation Problem Formulation

Here, the decision variable $x_{i,j}$ is the amount of goods shipped (i.e., some product) going from *i* to *j*, where *i* is the originating location and *j* is the destination. $c_{i,j}$ is the cost of shipping a unit of goods ($x_{i,j}$) from *i* to *j*, $a_i$ is the total amount of shipment from *i*, and $b_j$ is the total amount of shipment to *j*. The goal is to minimize the total costs ($c_{i,j}$) of shipping goods ($x_{i,j}$) from *i* to *j* (1) while meeting the requirements of supply (2) and demand (3). For this formulation, there is a restriction that $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$ where the supply is equal to demand, and if the $a_i$ and $b_j$ values are nonnegative integers (for all *i* and *j*), then all the basic feasible solutions, or extreme point values, have integer values (Gass 1985). This indicates the transportation problem can be solved as a linear programming (LP) problem (i.e., it can be solved in polynomial time).

The formulation of the personnel-assignment matching problem can be derived from the transportation problem using Model 1 with slight modifications (Gass 1985):

$$max \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} x_{i,j} \qquad (4)$$

$$s.t. \ \sum_{j=1}^{n} x_{i,j} = 1 \qquad i = 1, 2, ..., m \qquad (5)$$

$$\sum_{i=1}^{m} x_{i,j} = 1 \qquad j = 1, 2, ..., n \qquad (6)$$

$$x_{i,j} \geq 0 \qquad \forall \ i, j.$$

Model 2. Personnel-Assignment Matching Problem Formulation

Here, the decision variable $x_{i,j}$ is the assignment of individual *i* to job *j* and $c_{i,j}$ is the value of assigning *i* to *j* where the objective (4) is to maximize the total value of assigning *i* to *j*. The $a_i$ and $b_j$ values in constraints (2) and (3) from Model 1 are now equal to 1 where constraint (5) denotes that individual *i* must only fill one job *j* and constraint (6) denotes that job *j* must be filled by one individual *i*. $c_{i,j}$ denotes the value of individual *i* assigned to job *j*. Although the $x_{i,j}$ values in the personnel-assignment matching problem are 0 or 1, recall that if the right-hand side (RHS) values are nonnegative integers, then all the basic feasible solutions have integer values (Gass 1985). Thus, the binary condition for $x_{i,j}$ is omitted in Model 2.

An easy way to visualize the personnel-assignment matching problem is with a bipartite graph with a set of disjoint vertices and edges. An assignment matching problem is viewed as a bipartite graph with a set of vertices and edges, $G = (V_1, V_2, E)$, where the set of vertices are disjoint such that $V = (V_1 \cup V_2)$ and every edge *E* has one endpoint in $V_1$ and the other endpoint in $V_2$ (Wolsey 1998). Figure 1 is a representation of a bipartite graph where $V_1$ represents a set individuals and $V_2$ represents a set of jobs and edges represent potential assignments with cost (or value) $c_{i,j}$ of assigning an individual *i* to a job *j* such that individuals are only assigned to jobs (or vice versa) and not individuals to individuals or jobs to jobs:

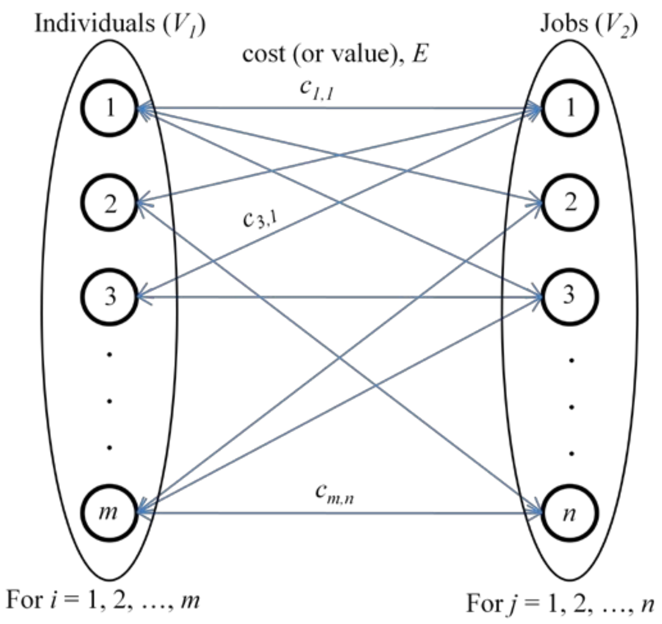


Figure 1. Bipartite Graph Representation

In a cost driven instance, the objective is to minimize the total cost (e.g., objective (1)). In a value driven instance, the objective is to maximize the total value (e.g., objective (4)).

Modeling assignment matching problems using bipartite graphs are generally preferred because they are well known problems for which many polynomial algorithms exist (i.e., many problems are either a form of an assignment matching problem or an extension of an assignment matching problem with additional constraints) (Caseau and Laburthe 2000). The most common way of formulating the bipartite assignment matching problem is as follows (Goemans 2009):

$$max \,(\text{or } \; min) \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} \qquad (7)$$

$$s.t. \; \sum_{j=1}^{n} x_{i,j} = 1 \qquad i \in V_1 = 1, 2, ..., m \qquad (8)$$

$$\sum_{i=1}^{m} x_{i,j} = 1 \qquad j \in V_2 = 1, 2, ..., n \qquad (9)$$

$$x_{i,j} \geq 0 \qquad \forall i, j.$$

Model 3. Bipartite Assignment Matching Problem Formulation

Here, $x_{i,j}$ is the assignment of individual *i* to job *j* (i.e., $x_{i,j} = 1$ if individual *i* is assigned to job *j* and 0 otherwise). Even though $x_{i,j}$ has a value of 0 or 1, recall that if the RHS of the constraints are nonnegative integers, then all the basic feasible solutions have integer values (Gass 1985), so a binary condition for $x_{i,j}$ is not necessary in the model. $c_{i,j}$ can be considered as the 'weight' of assigning individual *i* to job *j*. The 'weight' can be thought of as a cost (e.g., relocation costs for assigning individual *i* to job *j*) or as profit/value (e.g., what qualifications the individual can bring such as experience, education, skill sets, etc.). With the weight equal to one, the goal is to maximize the number of assignments of assigning individuals to jobs (7) while meeting the requirements that an individual must only fill one job (8) and that a job must be filled by one individual (9). With the weight equal to a value other than one, the goal is to minimize (or maximize) the total cost (or value) while meeting the same requirements. Note that in cases where the number of individuals and jobs are not necessarily equal to each other and the cases where not all individuals or jobs are filled, the equalities in constraints (5), (6), (8), and (9) can be replaced by inequalities (specifically $\leq$).

**Assignment Matching Problem Previous Research**

Note that for this sub-section, individuals represent candidates, persons, sailors, airmen, applicants, or faculty and personnel.

Assignment matching problems are used in many applications. Colucci et al. (2004) introduce the task assignment of individual profiles problem (TAP) of skill-matching in an organization using a weighted bipartite graph. Using personal profiles of individuals and an equal number of various tasking assignments, their algorithm determines a 'value' for each individual for each task. The value of the individual (defined as the suitability of an individual to a task) may be different depending on the task. The TAP formulation is identical to Model 2 where $c_{i,j}$ is the suitability of individual *i* to task *j* (Colucci, et al. 2004).

Hsieh et al. (1995) introduce an extension of the bipartite weighted matching problem (BWMP) called the bipartite weighted matching with penalty problem (BWMPP) where the graph, *G* = (*V*, *U*, *V* x *U*), is a weighted complete bipartite graph (i.e., each endpoint in *V* has an endpoint for every element in *U* and vice versa) and *V* = $(v_1, v_2, ..., v_m)$ and *U* = $(u_1, u_2, ..., u_n)$. Let *V* be the set of individuals and *U* be the set of tasks. Using the weight ($w_{i,j}$) of the respective edge ($v_i, u_j$) with penalty $s_i$ (for an unassigned vertex $v_i$) and penalty $t_j$ (for an unassigned vertex $u_j$), an assignment can be found that minimizes the sum of the weights and penalties. Hsieh et al. (1995) define the problem as follows:

$$min \sum_{i=1}^{m}\sum_{j=1}^{n} w_{i,j}x_{i,j} + \sum_{i=1}^{m}(1-\sum_{j=1}^{n}x_{i,j})s_i + \sum_{j=1}^{n}(1-\sum_{i=1}^{m}x_{i,j})t_j \qquad (10)$$

$$s.t. \sum_{i=1}^{m} x_{i,j} = 0 \text{ or } 1 \qquad j = 1, 2, ..., n \qquad (11)$$

$$\sum_{j=1}^{n} x_{i,j} = 0 \text{ or } 1 \qquad i = 1, 2, ..., m \qquad (12)$$

$$x_{i,j} \in \{0,1\} \qquad \forall\ i, j.$$

Model 4. Bipartite Weighted Matching with Penalty Problem Formulation

Here, $x_{i,j}$ is the assignment of individual *i* to task *j* and $w_{i,j}$ is the weight of the assignment, $s_i$ is the penalty for an unassigned individual, and $t_j$ is the penalty for an unassigned task. The objective is to find an assignment that minimizes the sum of the weights of the edges and the penalties of the vertices (10) while meeting the requirements that a task is filled (or not filled) by one individual (11) and that an individual can fill (or cannot fill) one task (12) (Hsieh, Ho and Fan 1995).

Garrett et al. (2005) introduce the sailor assignment problem (SAP) where the objective is to maximize the desirability of the assignment of individuals to jobs while minimizing costs. Every couple of years, individuals are required to change jobs. The challenge for the Navy is to ensure an assignment that keeps the individual happy and meets fleet requirements while minimizing costs for those assignments. If the cost is too much, the Navy cannot maintain its other priorities and if too much attention is focused on cost, there is the possibility that individuals will not be happy with their assignments which could result in a decrease in morale, retention, recruitment, etc. Similar to earlier models, Garrett et al. (2005) define the problem as follows:

$$
\begin{aligned}
& max \sum_{i=1}^{N}\sum_{j=1}^{M} F_{i,j} d_{i,j} && (13) \\
& s.t. \sum_{i=1}^{N} d_{i,j} \leq 1 \qquad \forall\, j \in \{1, 2,..., M\} && (14) \\
& \sum_{j=1}^{M} d_{i,j} \leq 1 \qquad \forall\, i \in \{1, 2,..., N\} && (15) \\
& d_{i,j} \in \{0,1\} \qquad \forall\, i, j.
\end{aligned}
$$

Model 5. Sailor Assignment Problem Formulation

Here, $d_{i,j}$ is the assignment of individual *i* to job *j* and $F_{i,j}$ is the 'fitness' of assigning individual *i* to job *j*. 'Fitness' is defined as all the relevant information to determine the desirability of the assignment. The objective is to maximize the fitness of individuals to jobs (13) while meeting the requirements that a job is filled by at most one individual (14) and that an individual can fill at most one job (15). Note the fitness score also takes into account the costs associated with job assignments meaning the higher the fitness score, the better assignment an individual to a job will be (Garrett, et al. 2005).

Kleeman and Lamont (2007) introduce the airman assignment problem (AAP), which is similar to the SAP. Both make use of the fitness of assigning an individual to a job (which includes associated penalties), but the AAP makes use of a hard constraint matrix where an individual either does or does not meet the hard constraints (e.g., security clearance, rank, and training for a job) and a soft constraint matrix which are penalty functions that are assessed when a job deviates from an ideal individual or the individual is not assigned to their ideal job. Also, the AAP has two objectives: the first determines how well each assignment meets the needs of the Air Force while satisfying the desires of the individual (met through the hard and soft constraint matrices) and the second takes into account the costs associated with moving from one job to the next. Because of the conflicting nature of both objectives, the result is a vector of solutions rather than a single point. Since both objectives are optimized, the goal is to minimize both objectives (16). Kleeman and Lamont (2007) define the problem as follows:

$$min \sum_{i=1}^{N}\sum_{j=1}^{M} F_{i,j} h_{i,j} a_{i,j} \qquad (16)$$

$$s.t. \ \ \mathrm{h}_{i,j} \in \{0,1\} \qquad \forall\ i,j \qquad (17)$$

$$\mathrm{a}_{i,j} \in \{0,1\} \qquad \forall\ i,j. \qquad (18)$$

Model 6. Airman Assignment Problem Formulation

Here, $F_{i,j}$ is the fitness of assigning individual *i* to job *j* (with associated penalties from the soft constraints), $h_{i,j}$ is the assignment of individual *i* to job *j* if individual *i* meets all hard constraints for job *j* (17), and $a_{i,j}$ is the assignment of individual *i* to job *j* (18) (Kleeman and Lamont 2007).

Phillips (1987) uses a multi-objective approach by introducing a weighting function for aggregating pre-emptive criteria for multi-objective assignment problems to assign individuals to parking lots at the University of Texas at Austin. Each individual is grouped into categories (e.g., ‘Deans’ with a value of 1 is the highest category through ‘all others’ with a value of 7, the lowest category) and each individual rank orders their preferences for parking lots. The objective is to minimize the total preference rating for each category preemptively (i.e., give each individual their best possible parking lot choice subject to their category). Phillips' (1987) model found a maximum set of assignments using preference ratings from each individual, weighted by the category of the individual, to solve the model as a single aggregate objective function (Phillips 1987).

Cimen (2001) introduces a model for the Turkish Armed Forces personnel assignment system (TAF). Like the AAP and Phillips (1987) model, the TAF uses a multi-objective model with preemptive goal programming to determine the optimal set of assignments. Cimen (2001) uses preemptive goal programming to solve three objectives: the organizational, career, and personal preference objectives. By solving each objective separately and obtaining the upper and lower bounds, initial RHS 'guesses' are determined as the side constraints in the main problem. If the first objective function is the most important, the RHS 'guesses' for the other objectives are determined by decreasing the upper bounds for the second and third objective functions which increases the first objective function value. Desired levels of tradeoffs are then found by ensuring the second and third objective function values are restricted by their respective lower bounds while trying to reach the first objective function value upper bound (Cimen 2001).

## Research Contribution

Note that for this sub-section and the remainder of this research, recruits represent individuals and Classes represent categories unless otherwise specified.

The previous works, aforementioned, have similarities and differences to this research. The TAP can be viewed as assigning values to recruits based upon the Class (or Classes) the recruit belongs to where the objective is to maximize the total value of the assignments (or minimize the low suitability of assignments, which can also be viewed as maximizing the high suitability of the assignments). However, this research differs by presenting a penalty in the objective function for not assigning high valued recruits based on the Class (or Classes).

The BWMPP can be viewed as assigning values to recruits based upon a Class (or Classes) the recruit belongs to. A penalty is assessed when a high valued recruit is not assigned to a job just as a penalty is assessed for an unassigned vertex in *V* and *U*, like the BWMPP. However, in this research, penalties are only assessed for feasible assignments that are unassigned (depending on the weights for each Class), whereas the BWMPP assesses a penalty for every vertex that is unmatched. Also, unlike the BWMPP, this research uses a multi-objective approach to determine the penalty assessed depending on the weight of the Class (or Classes).

The SAP can also be viewed as assigning values to recruits based upon a Class (or Classes) the recruit belongs to, which can be thought of as the fitness of the recruit. The objective maximizes the total value (or fitness) of the assignments. However, this research presents a different approach to deal with ‘cost’ as a penalty, which, in the case of the SAP, is incorporated in the fitness function. ‘Cost’ in this research is similar to focusing on a Class (or Classes). Depending on which Class (or Classes) is focused on can determine the final outcome of the overall assignments. For example, if the first Class is the primary focus, then a penalty is assessed and reflected in the overall score if jobs are not assigned to recruits who belong in the first Class. A higher weighting could result in more assignments for that Class.

The AAP, like the SAP, can be viewed as assigning values to recruits based upon a Class (or Classes) the recruit belongs to, which can be thought of as the fitness of the recruit. The soft constraints can be thought of as the recruit’s preferences, which add positive value to the value function. The hard constraints can be thought of as recruits who are qualified for hard-to-fill jobs in this research. The objective for the AAP is to assign jobs to recruits that meet the needs of the Air Force along with their desirability while minimizing cost. However, this research presents an objective that maximizes the total value of ‘quality’ assignments (where quality means meeting requirements for various Classes) by preemptively maximizing the number of quality recruits who can fill hard-to-fill jobs (a hard constraint) and then, depending on the Class focus, maximize the number of quality recruits to fill the remaining jobs, using a multi-objective approach.

This multi-objective approach is similar to what Phillips (1987) presents by introducing a weighting function for aggregating pre-emptive criteria for multi-objective assignment problems. This research also takes into account the value and preferences of recruits. However, this research assesses weighted penalties for not assigning high valued recruits depending on the focus of a Class (or Classes) using a multi-objective approach.

The TAF can be viewed as using preemptive goal programming to find the maximum number of recruits that can fill hard-to-fill jobs and then solving the next set of objectives afterwards. However, Cimen (2001) proposes that all three objectives are found using preemptive goal programming where this research uses preemptive goal programming for the first objective (i.e., hard-to-fill jobs) but uses a multi-objective approach to find desirable bounds for the Class objectives (i.e., ethnicity and QT Scores). Also, Cimen (2001) proposes to use degradation factors to ensure the hierarchical structure is protected among the objectives where this research uses deviational variables to ensure feasibility among the constraints for the various Class objectives, penalizing the objective function depending on the weights for the Class (or Classes).

The previous research presented on the assignment matching problems and variations of the assignment matching problems enables this research to focus on new methods of finding optimal solutions.  Specifically, this research finds an optimal set of assignments that maximizes the total value of the assignments while taking into consideration the Classes (i.e., ethnicity and QT Scores), and uses a weighting structure to articulate which Class (or Classes) to focus on. This method is an extension from all previous research presented with particular reference to the works of Kleeman and Lamont (2007), Phillips (1987), and Cimen (2001).

The next section presents the methodology with the model formulations and model verification.

# III. Methodology

This section explains the methodology that is used to solve the 369th Recruiting Squadron problem. First is an explanation of the methods that are used, followed by the various sets, parameters, and decision variables for the respective models. This is then followed with an explanation of the objective functions and constraints. Finally, the actual models are shown with the verification.

## Background

Note that for the remainder of this research, Classes represent the 369th goals/targets (i.e., Hispanics, African Americans, and AFQT scores $\geq 50$).

Recall the motivating problem for this research involves assigning new USAF recruits from the 369th Recruiting Squadron to available jobs in any given month. The solution approach uses two multi-criteria optimization techniques: the elastic constraint method and lexicographic optimization (Ehrgott 2005). The elastic constraint method is used to solve multiple objectives by choosing a single objective to minimize (or maximize), while the remaining objectives are used as constraints; this method relaxes the constraints and allows the respective constraints to be violated, penalizing any violation in the objective function which allows for trade-offs between multiple objectives (Ehrgott and Ryan 2002). For the 369th, the objective is to maximize the total value of all assignments while meeting certain requirements for each Class. Lexicographic optimization takes into consideration the order in which objectives are solved (Ehrgott 2005). For the 369th, the first task is to determine the maximum number of assignments for hard-to-fill jobs. The second task is to determine the maximum number of assignments for each Class. The last task uses the determined information to solve the main problem of maximizing the total value of all assignments. In order to fully understand the model formulations, the sets, parameters, and decision variables are now explained.

**Model Formulation Sets, Parameters, and Decision Variables**

The models present four different sets. $R$ is the set of recruits in a given month (i.e., $R = \{1, 2,\ldots, n\}$). $C$ is the set of Classes (i.e., $C = \{1, 2,\ldots, k\}$). $J$ is the set of jobs in a given month (i.e., $J = \{1, 2,\ldots, m\}$). $H$ is the set of hard-to-fill jobs in a given month where $H \subseteq J$.

There are several parameters which constitute the majority of the three models. $D_{r,j}$ represents the availability of recruit $r \in R$ to fill job $j \in J$. Each recruit provides a window of availability during which they will accept a job that is available during a certain timeframe. If a recruit's window of availability falls within the timeframe for the job, then the recruit is considered available to fill the job. $D_{r,j}$ has a value of 1 if recruit $r$ is available to fill job $j$ or $D_{r,j}$ has a value of 0 otherwise. $P_{r,j}$ represents the recruit's top five job preferences. Each recruit is given the option to list up to five top-job preferences that they are qualified to perform, along with additional job preferences in the event the recruit is not assigned to any of their top five choices. $P_{r,j}$ has a value of 1 if recruit $r$ prefers job $j$ (i.e., one of the top five job preferences) or $P_{r,j}$ has a value of 0 otherwise (i.e., one of the additional job preferences). $\text{Rec}_{r,c}$ represents a recruit $r \in R$ belonging to a Class $c \in C$. $\text{Rec}_{r,c}$ has a value of 1 if recruit $r$ belongs to Class $c$ or $\text{Rec}_{r,c}$ has a value of 0 otherwise. A Class can represent many areas such as physical fitness levels, a particular ethnicity, test scores, a particular height, or visual acuity. Specifically, for the 369th problem, there are three different Classes, $c$ = 1, 2, or 3. $\text{Rec}_{r,1}$ represents a Hispanic recruit and has a value of 1 if recruit $r$ is Hispanic or has a value of 0 otherwise. $\text{Rec}_{r,2}$ represents an African American recruit and has a value of 1 if recruit *r* is African American or has a value of 0 otherwise. $\text{Rec}_{r,3}$ represents a recruit with an AFQT score $\geq 50$ and has a value of 1 if recruit *r* has an AFQT score $\geq 50$ or has a value of 0 otherwise. Depending on the number of Classes, job preferences, and what the user defines as important, a recruit's value (or quality) is determined. $v_{r,j}$ represents the value of recruit *r* assigned to job *j* where $v_{r,j} = \sum_{c \in C} \text{Rec}_{r,c} + P_{r,j} + 1$. For instance in the 369th problem, if recruit *r* is Hispanic, African American, has an AFQT score $\geq 50$, and prefers to fill jo b *j*, then the recruit's value would be equal to 5. In a worst case scenario, the recruit's value is equal to 1; this ensures every recruit has a chance to fill a job. $w_c$ represents the weight for a given Class *c*. Depending on the user requirements, some Classes may be given a higher weight to focus on trying to create more assignments for that particular Class. Specifically, for the 369th problem, $0 \leq w_1,\ w_2,\ w_3 \leq |C|$, where $|C|$ is the maximum number of Classes in the problem instance. *F* represents the maximum number of assignments for all recruits in *R* assigned to hard-to-fill jobs in *H*. $T_c$ represents the maximum number of assignments for all recruits in *R*, who belong to a Class in *C*, assigned to jobs in *J*, for all Classes in *C*. Specifically, for the 369th problem, $T_1$ represents the maximum number of assignments for recruits that are Hispanic who are assigned to jobs in *J*, $T_2$ represents the maximum number of assignments for recruits that are African American who are assigned to jobs in *J*, and $T_3$ represents the maximum number of assignments for recruits with an AFQT score $\geq 50$ who are assigned to jobs in *J*.

The primary decision variable, $E_{r,j}$, represents the assignment between recruit *r* and job *j*. $E_{r,j}$ has a value of 1 if recruit *r* is assigned to job *j* or $E_{r,j}$ has a value of 0 otherwise. Note that because the Main Problem Formulation is totally unimodular (refer to Appendix E for detailed information), the condition for $E_{r,j}$ does not need to be binary which means the problem can be solved as a LP. $s_c$ represents the slack (or deviation from the target) for a particular Class *c*, and thus, $s_c = \max(0,\ T_c - \sum_{r \in R} \sum_{j \in J} \text{Rec}_{r,c} D_{r,j} E_{r,j})$. Specifically, for the 369th problem, $s_1$ represents the deviation from the target for the Hispanic Class, $s_2$ represents the deviation from the target for the African American Class, and $s_3$ represents the deviation from the target for the AFQT score $\geq 50$ Class.

**Model Formulation Sets, Parameters, and Decision Variables Summarization**

Table 1 through Table 3 summarizes the sets, parameters, and decision variables of the model formulation:

Table 1. Sets in the Model Formulation

| | |
|---|---|
| $R = \{1, 2, \ldots, n\}$ | The set of recruits in a given month |
| $C = \{1, 2, \ldots, k\}$ | The set of Classes |
| $J = \{1, 2, \ldots, m\}$ | The set of jobs in a given month |
| $H = \{1, 2, \ldots, h\}$ | The set of hard-to-fill jobs in a given month where $H \subseteq J$ |

Table 2. Parameters in the Model Formulation

| | |
|---|---|
| $D_{r,j}$ | 1 if recruit $r \in R$ is available to fill job $j \in J$, 0 otherwise |
| $P_{r,j}$ | 1 if recruit $r \in R$ prefers job $j \in J$ (i.e., one of the top five preferences), 0 otherwise (i.e., one of the additional preferences) |
| $\text{Rec}_{r,c}$ | 1 if recruit $r \in R$ belongs in Class $c \in C$, 0 otherwise |
| $v_{r,j}$ | Value of recruit $r \in R$ assigned to job $j \in J$ where $v_{r,j} = \sum_{c \in C} \text{Rec}_{r,c} + P_{r,j} + 1$ and $1 \leq v_{r,j} \leq 2 + \lvert C \rvert$ |
| $w_c$ | Weight for a given Class $c \in C$ where $0 \leq w_c \leq \lvert C \rvert$ |
| $F$ | Maximum number of assignments for all recruits in *R* assigned to hard-to-fill jobs in *H* |
| $T_c$ | Maximum number of assignments for all recruits in *R* assigned to jobs in *J* for each Class $c \in C$ |

Table 3. Decision Variables for the Model Formulation

| | |
|---|---|
| $E_{r,j}$ | The assignment of recruit $r \in R$ to job $j \in J$ where $E_{r,j} \in \{0,1\}$ |
| $s_c$ | Slack (or deviation from the target) for a particular Class $c \in C$ where $s_c \geq 0$ |

**Objective Function Formulations**

The main objective for the 369th assignment matching problem is to maximize the total value of the assignments (with quality recruits) subject to certain requirements. This is accomplished with lexicographic optimization, which maintains the preemptive nature of the research problem, using two Sub-problems within the Main Problem Formulation. Solving the objective function for Sub-problem 1 focuses on maximizing the total number of assignments for recruits to hard-to-fill jobs:

$$max\ F = \sum_{r \in R} \sum_{j \in H} D_{r,j} E_{r,j}. \qquad (19)$$

Objective function (19) maximizes the total number of assignments of recruits in *R* that are available to fill hard-to-fill jobs in *H*. After solving objective function (19), the next step is to solve the objective function for Sub-problem 2, which focuses on maximizing the total number of assignments for recruits in *R*, who belong to a Class in *C*, to jobs in *J*:

$$max\ T_c = \sum_{r \in R} \sum_{j \in J} \mathrm{Rec}_{r,c} D_{r,j} E_{r,j} \qquad \forall\ c \in C. \qquad (23)$$

Objective function(s) (23) maximizes the total number of assignments of recruits in *R*, who belong to a Class in *C*, available to fill jobs in *J*. After solving objective function(s) (23), the last step is to use the acquired information for *F* and $T_c$ for each Class in *C* and solve the objective function in the Main Problem Formulation which focuses on maximizing the total value of assignments with quality recruits:

$$max \sum_{r \in R} \sum_{j \in J} v_{r,j} D_{r,j} E_{r,j} - \sum_{c \in C} w_c s_c. \qquad (27)$$

Objective function (27) maximizes the total value of assignments with quality recruits in *R* to fill jobs in *J*.

**Constraint Formulations**

The first set of constraints for Sub-problem 1 makes sure that hard-to-fill job *j* is filled by at most one recruit in *R*:

$$\sum_{r \in R} D_{r,j} E_{r,j} \leq 1 \qquad \forall\, j \in H. \qquad (20)$$

Constraint (20) is summed over every recruit in *R* that is available to fill hard-to-fill job *j*. The second set of constraints for Sub-problem 1 makes sure that recruit *r* can fill at most one hard-to-fill job in *H*:

$$\sum_{j \in H} D_{r,j} E_{r,j} \leq 1 \qquad \forall\, r \in R. \qquad (21)$$

Constraint (21) is summed over every hard-to-fill job *j* that is available for recruit *r* to fill. The last set of constraints for Sub-problem 1 represents the decision variable:

$$E_{r,j} \in \{0,1\} \qquad \forall\, r \in R,\, j \in H. \qquad (22)$$

Constraint (22) states that the decision variable, $E_{r,j}$, is binary. However, according to Gass (1985), all basic feasible solutions have integer values (Gass 1985). This indicates that Sub-problem 1 can be solved as a LP.

The constraints for Sub-problem 2 are very similar to the constraints in Sub-problem 1 (i.e., constraints (20), (21), and (22)), with the exception that hard-to-fill jobs *j* are now referred to any jobs $j \in J$. The first set of constraints for Sub-problem 2 makes sure that job *j* is filled by at most one recruit in *R*:

$$\sum_{r \in R} D_{r,j} E_{r,j} \leq 1 \qquad \forall\, j \in J. \qquad (24)$$

Constraint (24) is summed over every recruit in *R* that is available to fill job *j*. The second set of constraints for Sub-problem 2 makes sure that recruit *r* can fill at most one job in *J*:

$$\sum_{j \in J} D_{r,j} E_{r,j} \leq 1 \qquad \forall\ r \in R. \qquad (25)$$

Constraint (25) is summed over every job in *J* that is available for recruit *r* to fill. The last set of constraints for Sub-problem 2 represents the decision variable:

$$E_{r,j} \in \{0,1\} \qquad \forall\ r \in R,\ j \in J. \qquad (26)$$

Constraint (26) states that the decision variable, $E_{r,j}$, is binary. Similar to Sub-problem 1, all basic feasible solutions have integer values (Gass 1985). Thus, Sub-problem 2 can also be solved as a LP.

The constraints from Sub-problem 2 are also prevalent in the constraint set for the Main Problem Formulation:

$$\sum_{r \in R} D_{r,j} E_{r,j} \leq 1 \qquad \forall\ j \in J \qquad (28)$$

$$\sum_{j \in J} D_{r,j} E_{r,j} \leq 1 \qquad \forall\ r \in R \qquad (29)$$

$$E_{r,j} \in \{0,1\} \qquad \forall\ r \in R,\ j \in J. \qquad (32)$$

Specifically, the first (constraint (28)), second (constraint (29)), and fifth (constraint (32)) constraint sets in the Main Problem Formulation are the same as the constraint sets from Sub-problem 2, respectively, and serve the same purpose. After solving for *F* in Sub-problem 1, *F* becomes the RHS for the third constraint set in the Main Problem Formulation:

$$\sum_{r \in R} \sum_{j \in H} D_{r,j} E_{r,j} \geq F \qquad \text{(Sub-problem 1).} \qquad (30)$$

Constraint (30) ensures that the maximum number of assignments of recruits in *R*, available to fill hard-to-fill jobs in *H*, is attained. After solving for $T_c$ in Sub-problem 2 for each Class in *C*, $T_c$ becomes the RHS for the fourth constraint set in the Main Problem Formulation:

$$\sum_{r \in R}\sum_{j \in J} \mathrm{Rec}_{r,c} D_{r,j} E_{r,j} + s_c \geq T_c \qquad \forall\ c \in C \text{ (Sub-problem 2).} \qquad (31)$$

Constraint (31) ensures that the maximum number of assignments of recruits in *R*, who belong to a Class in *C* that are available to fill jobs in *J*, is attained. This number also depends on the deviation from the target, $s_c$, for each Class in *C*. The last set of constraints for the Main Problem Formulation represents the non-negativity of the deviation from the target, $s_c$, for each Class in *C*:

$$s_c \geq 0 \qquad \forall\ c \in C. \qquad (33)$$

Constraint (33) ensures that the deviation from the target, $s_c$, for each Class in *C* will be a positive number; otherwise, a negative $s_c$ value for any Class in *C* in constraint (33) would make the Main Problem Formulation infeasible. Recall that because the Main Problem Formulation is totally unimodular, the binary condition for $E_{r,j}$ does not need to be imposed since the problem can be solved as a LP. Refer to Appendix E for detailed information on total unimodularity for the Main Problem Formulation.

Placing the constraints with the respective objective functions results in the model formulation for Sub-problem 1, Sub-problem 2, and the Main Problem Formulation, respectively:

$$max\ F = \sum_{r \in R} \sum_{j \in H} D_{r,j} E_{r,j} \qquad (19)$$

*subject to*

$$\sum_{r \in R} D_{r,j} E_{r,j} \le 1 \qquad \forall\ j \in H \qquad (20)$$

$$\sum_{j \in H} D_{r,j} E_{r,j} \le 1 \qquad \forall\ r \in R \qquad (21)$$

$$E_{r,j} \in \{0,1\} \qquad \forall\ r \in R,\ j \in H. \qquad (22)$$

Model 7. Sub-problem 1 for Hard-to-fill Jobs

$$max\ T_c = \sum_{r \in R} \sum_{j \in J} \mathrm{Rec}_{r,c} D_{r,j} E_{r,j} \qquad \forall\ c \in C \qquad (23)$$

*subject to*

$$\sum_{r \in R} D_{r,j} E_{r,j} \le 1 \qquad \forall\ j \in J \qquad (24)$$

$$\sum_{j \in J} D_{r,j} E_{r,j} \le 1 \qquad \forall\ r \in R \qquad (25)$$

$$E_{r,j} \in \{0,1\} \qquad \forall\ r \in R,\ j \in J. \qquad (26)$$

Model 8. Sub-problem 2 for Different Classes

$$max\ \sum_{r \in R} \sum_{j \in J} v_{r,j} D_{r,j} E_{r,j} - \sum_{c \in C} w_c s_c \qquad (27)$$

*subject to*

$$\sum_{r \in R} D_{r,j} E_{r,j} \le 1 \qquad \forall\ j \in J \qquad (28)$$

$$\sum_{j \in J} D_{r,j} E_{r,j} \le 1 \qquad \forall\ r \in R \qquad (29)$$

$$\sum_{r \in R} \sum_{j \in H} D_{r,j} E_{r,j} \ge F \qquad \text{(Sub-problem 1)} \qquad (30)$$

$$\sum_{r \in R} \sum_{j \in J} \mathrm{Rec}_{r,c} D_{r,j} E_{r,j} + s_c \ge T_c \qquad \forall\ c \in C \text{ (Sub-problem 2)} \qquad (31)$$

$$E_{r,j} \in \{0,1\} \qquad \forall\ r \in R,\ j \in J \qquad (32)$$

$$s_c \ge 0 \qquad \forall\ c \in C. \qquad (33)$$

Model 9. Main Problem Formulation

## Verification

To verify the matching algorithm produced the correct results when LINGO solved each problem instance, small randomly generated instances were used to compare the algorithm output against manual implementation. These instances were varied between: the total number of recruits and jobs (both $\leq 10$), recruits of different Classes, recruit and job availability, and job types (i.e., hard-to-fill jobs vs. non hard-to-fill jobs). The matching algorithm was used for each small instance where each small instance produced LINGO files with the corresponding Sub-problems for hard-to-fill jobs and each Class (if either existed) and the Main Problem Formulation. With the same small instances, the LP models were manually typed into respective LINGO files. For each LINGO file (matching algorithm generated and manually generated), assignments were compared to one another. If both results were equal in terms of objective functions and assignments, it would be considered successful and another small randomly generated instance would be created to make further comparisons. This procedure was performed at least 30 times and in every case, both the matching algorithm generated and manually generated LINGO files were equal to one another. Also, the matching algorithm generated and manually generated LINGO files were compared to each other to verify correct model formulation. In each case, both matching algorithm and manually generated LP model formulations were similar to each other. During the verification, it was assumed that because LINGO is a well-known, widely used and accepted optimization software package, the LINGO solver provided correct results with the given LP model formulations.

Application of this methodology is presented in the next section with testing, results, analysis, and validation.

# IV. Application, Results, and Analysis

This section explains the application, results, and analysis using the methodology from the previous section. First, the necessary inputs required for the models are explained. Next, the output that the matching algorithm provides is discussed. This is then followed with some background information to answer the research questions (previously mentioned in Section I). Finally, the validation of the matching algorithm is explained.

## Inputs, Outputs, and System Platform

For the 369th Recruiting Squadron model implementation, certain inputs are required to solve the three different models. These inputs are specified using Microsoft Excel. First, the availability of the recruit and job is required. This is given by $D_{r,j}$ which is defined as a $n$ x $m$ matrix of recruits and jobs; if true, a "1" is placed within the respective cell and a blank otherwise (i.e., if $D_{r,j}$ = 1, then a "1" is placed in the $r$th row of the $j$th column of $D$ or a blank if $D_{r,j}$ = 0). Second, the recruit's preference for a job is required. This is given by $P_{r,j}$ which is defined as another $n$ x $m$ matrix of recruits and jobs; if one of the recruit's top five preferences is present in the jobs for a given month, a "1" is placed within the respective cell and a blank otherwise (i.e., if $P_{r,j}$ = 1, then a "1" is placed in the $r$th row of the $j$th column of $P$ or a blank if $P_{r,j}$ = 0). Third, the recruit's Class is required. This is given by $\text{Rec}_{r,c}$ which is defined as a $n$ x $k$ matrix of recruits and Classes. If a recruit falls under any Class, a "1" is placed within the respective cell and a blank otherwise (i.e., if $\text{Rec}_{r,c}$ = 1, then a "1" is placed in the $r$th row of the $c$th column of Rec or a blank if $\text{Rec}_{r,c}$ = 0). For the 369th, this is a $n$ x 3 matrix of recruits and Classes 1, 2, and 3 (i.e., Hispanics, African American, and QT Score $\geq$ 50, respectively). Fourth, hard-to-fill jobs are required. Recall that hard-to-fill jobs in *H* are a subset of the jobs in *J* (i.e., the hard-to-fill jobs matrix is a subset of $D_{r,j}$). The hard-to-fill jobs is defined as another matrix of at most *n* x *m* recruits and hard-to-fill jobs; if true, a "1" is placed within the respective cell and a blank otherwise (i.e., if $D_{r,j}$ = 1, where *j* is a hard-to-fill job, then a "1" is placed in the *r*th row of the *j*th column of the hard-to-fill jobs matrix or a blank if $D_{r,j}$ = 0). Finally, weighting levels are used for each Class consideration. This allows the user to focus on a certain Class (or Classes) when the matching algorithm creates an optimal set of assignments. Note that although the matrices may be sparse, arrays are used to store the inputs in order to save time developing the model formulations. If memory or storage is an issue, there are alternative methods to store and manipulate the data.

Although there are several outputs created by the models, the final output is the only one the user is interested in given the inputs for the 369th models. The first set of outputs (Sub-problem 1) provides the maximum number of hard-to-fill jobs given $D_{r,j}$. This is a basic assignment matching formulation that maximizes the number of assignments of hard-to-fill jobs given one recruit can occupy at most one hard-to-fill job and one hard-to-fill job can be occupied by at most one recruit. The next set of outputs (Sub-problem 2) provides the maximum number of assignments of recruits who fall under certain Classes. Again, this is a basic assignment matching formulation that maximizes the number of assignments of Class 1 given at most one recruit can occupy one job and one job can be occupied by at most one recruit. This is the same for Classes 2 and 3. The solutions from the hard-to-fill jobs and from each Class are used in the final model (Main Problem Formulation) to create the final output the user is interested in. The output maximizes the total value of all recruit assignments while given the same constraints that one recruit can occupy at most one job and one job can be occupied by at most one recruit. This final output also meets the maximum number of hard-to-fill job assignments and minimizes the weighted deviation from the number of assignments for each Class with the given weighting levels. The final outputs are:

1) The specific recruit/job assignments.
2) The percentage of assignments for each Class compared to the total number of assignments.
3) The percentage of assignments for each Class compared to the maximum allowable assignments for each Class.
4) The total number of assignments made.
5) The maximum number of assignments each Class could achieve.
6) The total number of assignments the matching algorithm finds for each Class given the weighting levels.
7) The total number of jobs with the percentage of jobs assigned.

All tests and experiments were run using a PC with Windows XP Professional with Service Pack 3 as an operating system with an AMD Athlon 64 X2 Dual Core Processor 4800+ (2.49 GHz), and 1.87 GB of RAM. The optimal solutions were found using the Extended LINGO solver, version 11.0.0.20 (released 11 June 2008) with some changes to the default LINGO settings. Refer to Figure 2 and Figure 3 in Appendix B for detailed settings.

## Generated Instances

Due to the limited number of monthly instances the 369th could provide, randomly generated instances were used to produce statistical significance. To simulate what a typical month may look like, four 369th monthly instances were used to find low and high ranges. Using the four 369th monthly instances, the averages and standard deviations were found for the following categories:

1) Total number of recruits in a given month.
2) Total number of jobs in a given month.
3) Percentage of Hispanics in a given month.
4) Percentage of African Americans in a given month.
5) Percentage of recruits who had QT Scores $\geq 50$ in a given month.
6) Total number of hard-to-fill jobs in a given month.
7) Percentage of the availability of recruits and jobs ($D_{r,j}$) in a given month.
8) Percentage of the recruits' preferences ($P_{r,j}$) in a given month.

For the category ranges, the low values of each category above is equal to the lowest value of the category minus one standard deviation and the high values of each category above is equal to the highest value of the category plus one standard deviation. For example, the lowest value from the four 369th monthly instances for total recruits was 105, the highest value was 255, and the standard deviation for this category was 68.16 recruits. The floor of the subtraction of 105 and 68.16 is 36 recruits. The ceiling of the addition of 255 and 68.16 is 324 recruits. Therefore, the total number of recruits for each month varied randomly between 36 and 324. This same approach was used for the remaining categories. Table 4 displays the results from the four 369th monthly instances; these results were used to construct the low and high ranges for the various categories in order to generate random monthly instances:

Table 4. Actual 369th Monthly Results

| Categories | Dec 2008 | Feb 2009 | Mar 2009 | Apr 2009 |
|---|---|---|---|---|
| # Recruits | 105 | 241 | 255 | 217 |
| # Jobs | 93 | 109 | 114 | 103 |
| % Hispanic | 43.81 | 46.06 | 45.10 | 47.00 |
| % African Amer. | 12.38 | 8.71 | 6.27 | 7.37 |
| % QT Score ≥ 50 | 58.10 | 66.39 | 67.84 | 59.91 |
| # Hard-to-fill Jobs | 2 | 6 | 10 | 3 |
| % Availability | 5.56 | 8.82 | 8.28 | 9.48 |
| % Preferred | 8.32 | 8.72 | 6.44 | 6.65 |

Table 5 displays the averages, standard deviations, and the low/high ranges used to generate random monthly instances with the results from Table 4:

Table 5. Averages, Standard Deviations, Low/High Ranges from Table 4 Results

| Categories | Average | Standard Deviation | Low Range | High Range |
|---|---|---|---|---|
| # Recruits | 204.5 | 68.16 | 36 | 324 |
| # Jobs | 104.75 | 9.03 | 83 | 124 |
| % Hispanic | 45.49 | 1.37 | 42 | 49 |
| % African Amer. | 8.69 | 2.66 | 3 | 16 |
| % QT Score ≥ 50 | 63.06 | 4.78 | 53 | 73 |
| # Hard-to-fill Jobs | 5.25 | 3.59 | 0 | 14 |
| % Availability | 8.04 | 1.72 | 3 | 12 |
| % Preferred | 7.53 | 1.15 | 5 | 10 |

**Time Results and Analysis**

Recall that the current 369th Recruiting Squadron process of creating assignments is performed manually and can take up to two weeks to complete. Thus, to answer the question of how much time could be saved using the matching algorithm provided in this research versus the 369th's current method, randomly generated monthly instances for three groups were used. The first group is the Squadron Level (SL) size instances (from Table 5 above), the second group is the Multi-Squadron Level (MSL) size instances (a factor between 2-3 times the size of the first group), and the final group is the Air Force Level (AFL) size instances (limited to the amount of

CPU memory). Both the MSL and AFL size instances used the same range parameters as the SL size instances with the exceptions of the number of recruits, number of jobs, and number of hard-to-fill jobs. The number of hard-to-fill jobs (high range) for the MSL and AFL size instances was found using the percentage of hard-to-fill jobs in the SL size instances and multiplying that percentage by the high range from the MSL and AFL size instances. Table 6 displays the ranges used for the three groups:

Table 6. Ranges for the Three Groups

| Categories | SL (low) | SL (high) | MSL (low) | MSL (high) | AFL (low) | AFL (high) |
|---|---|---|---|---|---|---|
| # Recruits | 36 | 324 | 108 | 648 | 1300 | 1500 |
| # Jobs | 83 | 124 | 166 | 372 | 1000 | 1400 |
| % Hispanic | 42 | 49 | 42 | 49 | 42 | 49 |
| % African Amer. | 3 | 16 | 3 | 16 | 3 | 16 |
| % QT Score ≥ 50 | 53 | 73 | 53 | 73 | 53 | 73 |
| # Hard-to-fill Jobs | 0 | 14 | 0 | 42 | 0 | 158 |
| % Availability | 3 | 12 | 3 | 12 | 3 | 12 |
| % Preferred | 5 | 10 | 5 | 10 | 5 | 10 |

Using Excel's data analysis function, random numbers were generated using a uniform distribution with the low and high ranges for each category for the three groups. 2,000 instances were randomly generated for the first group, 1,500 instances were randomly generated for the second group, and 1,000 instances were randomly generated for the final group.

Table 7 through Table 9 displays the results for the computational times (in seconds) to create the LP model formulations using VBA, the computational times (in seconds) to solve the LP models with LINGO, and the total computational time (in seconds) using the combination of VBA and LINGO times for each group, respectively:

Table 7. Results from the Three Groups (VBA Times in seconds)

| Size | Average | Standard Deviation | 95% Confidence Interval |
|---|---|---|---|
| SL | 1.61 | 1.57 | (1.54, 1.68) |
| MSL | 9.95 | 8.35 | (9.53, 10.37) |
| AFL | 199.47 | 111.63 | (192.68, 206.50) |

Table 8. Results from the Three Groups (LINGO Times in seconds)

| Size | Average | Standard Deviation | 95% Confidence Interval |
|---|---|---|---|
| SL | 0.31 | 0.29 | (0.29, 0.32) |
| MSL | 1.34 | 0.84 | (1.29, 1.38) |
| AFL | 55.70 | 23.20 | (54.28, 57.16) |

Table 9. Results from the Three Groups (Total Times in seconds)

| Size | Average | Standard Deviation | 95% Confidence Interval |
|---|---|---|---|
| SL | 1.91 | 1.66 | (1.84, 1.98) |
| MSL | 9.05 | 7.02 | (10.85,11.74) |
| AFL | 255.17 | 125.49 | (247.32, 263.08) |

The averages and standard deviations are based upon the number of randomly generated instances for each group. However, because non-normal distributions were detected for the three groups for the three times reported (refer to Figure 4 through Figure 12 in Appendix B), a bootstrap method, using matrix laboratory's (MATLAB) bootstrap function, was used to find the respective confidence intervals. The bootstrap method is a technique that allows the estimation of a sample, regardless of its distribution (Varian 2005). In this case, because the data suggests a non-normal distribution, the bootstrap method (using 10,000 re-samples) was used to calculate the 95% confidence intervals for the three groups for the three times reported in Table 7 through Table 9.

Currently, the 369th manually assigns recruits and jobs each month and can take up to two weeks to come up with a feasible set of assignments. These assignments are not guaranteed to be optimal in terms of the user's end goal. Based on the respective results reported for the three groups, it is evident that the matching algorithm presented in this research is clearly faster at determining a feasible set of assignments that is also guaranteed to be optimal. To put it in perspective of man-hour savings per year, assume all times involved, with the exception of creating assignments, are the same for the matching algorithm and the 369th current method (e.g., getting user data before creating assignments and analyzing results after assignments are created). Now, assume the 369th takes five working days (approximately one week) to solve a problem instance using the ranges for the SL size instances and it takes one recruiter, on average, 5 man-hours per day. Over the one week period, 5x5 = 25 man-hours are spent to create a feasible set of assignments for each month. During the course of the year, approximately 25x12 (months) = 300 man-hours are spent trying to create feasible assignments that are not necessarily guaranteed to be optimal. According to the statistics this research provides, it would take, on average, 1.91 seconds (or 0.00053 hours) for the matching algorithm to provide an optimal set of assignments. Therefore, during the course of one year, approximately 1.91x12 (months) = 22.92 seconds would be spent to create an optimal set of assignments. That is a savings of approximately 299.99 man-hours per year. In an extreme case, make the same assumptions as stated above but using the ranges for the AFL size instances. During the course of one year, approximately 255.17x12 (months) = 3,062.04 seconds (or 0.85 hours) would be spent to create an optimal set of assignments using the matching algorithm provided in this research. This is still a savings of approximately 299.15 man-hours per year. Note that 25 man-hours spent to create a feasible set of assignments each month is a very conservative estimate for AFL size instances, which would most likely require a more significant amount of time.

There are limitations to using the matching algorithm. The amount of processing power can affect the time it takes to solve instances of the three groups. If a slower processor is used, it is expected that the results in Table 7 through Table 9 would be slower; faster times would be expected with a faster processor. The amount of physical memory that was used in this research limited the size of randomly generated AFL size instances; more physical memory would allow for larger AFL size instances. Refer to Appendix C for further information regarding CPU times to solve problem instances.

**Quantity vs. Quality of Assignments Results and Analysis**

The 369th Recruiting Squadron would like to assign as many recruits as possible to available jobs for any given month (subject to the total number of jobs available for that given month). Although the 369th would like to maximize the number of assignments, they are more interested in assigning quality recruits to jobs. If assigning quality recruits to jobs also maximizes the number of assignments, then maximizing the number of assignments would be an added bonus. Thus, to answer the question if finding the maximum number of assignments versus maximum value of assignments provides better results in terms of percentage of assignments for each Class, the same randomly generated instances from the first group (SL size instances) are used to conduct this test. All LP model formulations are created using the matching algorithm provided in this research; however, there are two Main Problem Formulations to capture the differences in quantity versus quality. The first Main Problem Formulation comes from Model 9 and the second Main Problem Formulation uses Model 9 but adds the constraint (*) of meeting the maximum number of assignments allowed:

$$max \sum_{r\in R}\sum_{j\in J} v_{r,j} D_{r,j} E_{r,j} - \sum_{c\in C} w_c s_c \qquad (27)$$

*subject to*

$$\sum_{r\in R}\sum_{j\in J} D_{r,j} E_{r,j} = M \qquad (*)$$

$$\sum_{r\in R} D_{r,j} E_{r,j} \le 1 \qquad \forall\, j \in J \qquad (28)$$

$$\sum_{j\in J} D_{r,j} E_{r,j} \le 1 \qquad \forall\, r \in R \qquad (29)$$

$$\sum_{r\in R}\sum_{j\in H} D_{r,j} E_{r,j} \ge F \qquad \text{(Sub-problem 1)} \qquad (30)$$

$$\sum_{r\in R}\sum_{j\in J} \text{Rec}_{r,c} D_{r,j} E_{r,j} + s_c \ge T_c \qquad \forall\, c \in C \text{ (Sub-problem 2)} \qquad (31)$$

$$E_{r,j} \in \{0,1\} \qquad \forall\, r \in R,\ j \in J \qquad (32)$$

$$s_c \ge 0 \qquad \forall\, c \in C. \qquad (33)$$

Model 10. Main Problem Formulation with Maximum Number of Matches

Model 10 maximizes the value of assignments with quality recruits in *R* to fill jobs in *J* but uses constraint (*) to ensure the maximum number of assignments allowed, *M*, is achieved. *M* is found by using Model 9, but eliminating the value parameter ($v_{r,j}$) in the objective function (**). Specifically:

$$max\ M = \sum_{r\in R}\sum_{j\in J} D_{r,j} E_{r,j} - \sum_{c\in C} w_c s_c \qquad (**)$$

*subject to*

$$\sum_{r\in R} D_{r,j} E_{r,j} \le 1 \qquad \forall\, j \in J \qquad (28)$$

$$\sum_{j\in J} D_{r,j} E_{r,j} \le 1 \qquad \forall\, r \in R \qquad (29)$$

$$\sum_{r\in R}\sum_{j\in H} D_{r,j} E_{r,j} \ge F \qquad \text{(Sub-problem 1)} \qquad (30)$$

$$\sum_{r\in R}\sum_{j\in J} \text{Rec}_{r,c} D_{r,j} E_{r,j} + s_c \ge T_c \qquad \forall\, c \in C \text{ (Sub-problem 2)} \qquad (31)$$

$$E_{r,j} \in \{0,1\} \qquad \forall\, r \in R,\ j \in J \qquad (32)$$

$$s_c \ge 0 \qquad \forall\, c \in C. \qquad (33)$$

Model 11. Maximum Matching Model

Model 11 finds the maximum number of assignments for recruits in *R* to fill jobs in *J*. In both model formulations (quantity and quality), a weighting of zero (i.e., $w_c$ = 0 for all Classes in *C*) was used. A weighting of zero preserves objectivity when maximizing both objective functions when capturing the differences in assignments between both model formulations (if any differences exist).

Table 10 through Table 12 displays the results obtained using the same 2,000 randomly generated instances from the first group (SL size instances) for the quantity of assignments, the quality of assignments, and the differences between both model formulations:

Table 10. Quantity of Assignments Results

| | Total Matches | Class 1 (Hispanics) | Class 2 (African Americans) | Class 3 (QT Scores ≥ 50) |
|---|---|---|---|---|
| Average | 94.89 | 62.36% | 14.57% | 77.51% |
| Std. Dev. | 20.48 | 14.24% | 7.90% | 12.28% |
| 95% C.I. | (94.00, 95.79) | (61.74%, 62.97%) | (14.22%, 14.92%) | (76.98%, 78.04%) |

Table 11. Quality of Assignments Results

| | Total Matches | Class 1 (Hispanics) | Class 2 (African Americans) | Class 3 (QT Scores ≥ 50) |
|---|---|---|---|---|
| Average | 94.87 | 62.95% | 14.69% | 77.89% |
| Std. Dev. | 20.48 | 14.79% | 8.05% | 12.50% |
| 95% C.I. | (93.98, 95.75) | (62.30%, 63.61%) | (14.33%, 15.04%) | (77.34%, 78.44%) |

Table 12. Difference between Quantity and Quality of Assignments Results

| | Total Matches | Class 1 (Hispanics) | Class 2 (African Americans) | Class 3 (QT Scores ≥ 50) |
|---|---|---|---|---|
| Average | 0.03 | 1.32% | 0.52% | 1.10% |
| Std. Dev. | 0.21 | 1.66% | 0.85% | 1.35% |
| 95% C.I. | (0.02, 0.04) | (1.24%, 1.39%) | (0.48%, 0.55%) | (1.04%, 1.16%) |

The averages and standard deviations are based upon the 2,000 randomly generated instances from the first group (SL size instances). However, because non-normal distributions were detected from the total number of assignments and the three Classes between the quantity, quality, and the differences between both model formulations (refer to Figure 13 through Figure 24 in Appendix B), a bootstrap method, using MATLAB's bootstrap function, was used to find the respective 95% confidence intervals using 10,000 re-samples. The percentages for each Class were found by taking the total number of assignments for each Class and dividing by the total number of assignments (e.g., using Table 11, of the 94.87 total matches, 62.95% were from Class 1, 14.69% were from Class 2, and 77.89% were from Class 3).

The quantity model formulation is expected to focus more on maximizing the number of assignments for each problem instance and the quality model formulation is expected to focus more on maximizing the number of assignments for each Class. Table 12 shows that with the 2,000 randomly generated instances, the quantity model formulation was slightly higher on average compared to the quality model formulation when it came to maximizing the number of assignments. Note that in approximately 97.80% of the 2,000 randomly generated instances, the total number of assignments using the quantity model formulation equaled the total number of assignments using the quality model formulation. Table 12 also shows that for each respective Class, the quantity model formulation was slightly lower on average compared to the quality model formulation when it came to maximizing the number of assignments for each Class. Although it appears that both model formulations are similar because the differences are very small, they are statistically different. Using a paired t-test to compare the differences in averages between the quantity and quality model formulations, the results indicate the two model formulations are statistically different (refer to Appendix B, Table 22 through Table 25). Because the 369th is more interested in assigning quality recruits to jobs, the quality model formulation is used to perform the analysis on the weighting sensitivity for each Class.

**Weighting Sensitivity Results and Analysis**

Recall that the 369th Recruiting Squadron is more interested in assigning quality recruits to jobs for any given month. The 369th also has yearly goals/targets to achieve for each Class (i.e., there are yearly goals/targets to meet (or exceed) certain percentages for each Class). Using appropriate weighting levels for each Class could result in meeting (or exceeding) the yearly goals/targets set for each Class. Thus, to answer the question of meeting (or exceeding) the user's defined percentage goals/targets for each Class using the matching algorithm provided in this research, different scenarios are compared to examine the sensitivity on the weighting levels for each Class using 30 different randomly generated instances (using the ranges for the SL size instances in Table 6) per scenario. Two different experiments were performed to capture the percentage of recruits from a certain Class that are assigned compared to the maximum number of assignments for that respective Class (actual assigned from a Class divided by the maximum assignments allowable for that respective Class) and to capture the percentage of recruits for each Class that are assigned compared to the total number of assignments (actual assigned from a Class divided by the total number of assignments). For the 369th, there are three Classes and the different scenarios use weighting levels of 0, 1, 2, and 3. Using three Classes and four weighting levels, 64 different scenarios were created to examine the sensitivity for every weighting level possibility. With 30 different randomly generated instances per scenario (at four weighting levels each), 64x30 = 1,920 total runs were performed for each experiment to capture the overall weighting sensitivity. JMP was used to create the designs of experiments (DOEs) to capture the sensitivity of the weighting levels for each Class.

The first experiment compared the percentage of recruits actually assigned from each Class versus the maximum number of recruits that could be assigned in each respective Class. Table 13 through Table 15 displays the weighting sensitivity from the 64 scenarios with the weighting levels for Class 1, 2, and 3, respectively:

Table 13. JMP Parameter Estimates for Class 1: Hispanics (Max Possible)

| **Term** | **Estimate** | **Std Error** | **t Ratio** | **Prob>\|t\|** |
|---|---|---|---|---|
| Intercept | 0.866993 | 0.006067 | 142.89 | 0.0000* |
| w1 | 0.0493301 | 0.002145 | 23.00 | <.0001* |
| w2 | -0.003388 | 0.002145 | -1.58 | 0.1144 |
| w3 | -0.039323 | 0.002145 | -18.33 | <.0001* |
| (w1-1.5)*(w2-1.5) | -0.003981 | 0.001919 | -2.07 | 0.0382* |
| (w1-1.5)*(w3-1.5) | -0.00472 | 0.001919 | -2.46 | 0.0140* |
| (w2-1.5)*(w3-1.5) | 0.0061084 | 0.001919 | 3.18 | 0.0015* |
| (w1-1.5)*(w2-1.5)*(w3-1.5) | 0.0003339 | 0.001716 | 0.19 | 0.8458 |

Table 14. JMP Parameter Estimates for Class 2: African Americans (Max Possible)

| **Term** | **Estimate** | **Std Error** | **t Ratio** | **Prob>\|t\|** |
|---|---|---|---|---|
| Intercept | 0.8443679 | 0.007557 | 111.74 | 0.0000* |
| w1 | -0.014565 | 0.002672 | -5.45 | <.0001* |
| w2 | 0.0606193 | 0.002672 | 22.69 | <.0001* |
| w3 | -0.020863 | 0.002672 | -7.81 | <.0001* |
| (w1-1.5)*(w2-1.5) | -0.003323 | 0.00239 | -1.39 | 0.1645 |
| (w1-1.5)*(w3-1.5) | 0.0211528 | 0.00239 | 8.85 | <.0001* |
| (w2-1.5)*(w3-1.5) | -0.004403 | 0.00239 | -1.84 | 0.0655 |
| (w1-1.5)*(w2-1.5)*(w3-1.5) | -0.009338 | 0.002137 | -4.37 | <.0001* |

Table 15. JMP Parameter Estimates for Class 3: QT Scores (Max Possible)

| **Term** | **Estimate** | **Std Error** | **t Ratio** | **Prob>\|t\|** |
|---|---|---|---|---|
| Intercept | 0.8938322 | 0.00544 | 164.30 | 0.0000* |
| w1 | -0.035135 | 0.001923 | -18.27 | <.0001* |
| w2 | -0.005095 | 0.001923 | -2.65 | 0.0081* |
| w3 | 0.0437538 | 0.001923 | 22.75 | <.0001* |
| (w1-1.5)*(w2-1.5) | 0.0050627 | 0.00172 | 2.94 | 0.0033* |
| (w1-1.5)*(w3-1.5) | 0.0021218 | 0.00172 | 1.23 | 0.2176 |
| (w2-1.5)*(w3-1.5) | -0.00405 | 0.00172 | -2.35 | 0.0187* |
| (w1-1.5)*(w2-1.5)*(w3-1.5) | -0.000532 | 0.001539 | -0.35 | 0.7295 |

The tables above estimate the amount of trade-off that could be expected for an increase in one weighting level for each Class. For example, in Table 13, if the weight for Class 1 increases by one weighting level, the estimate for the percent of assignments from Class 1 would increase approximately 4.93%, Class 2 would decrease approximately 1.46%, and Class 3 would decrease approximately 3.51%. Note this example is solely focusing on Class 1 (i.e., the weighting levels for Classes 2 and 3 are zero, or $w_2 = w_3 = 0$).

The first experiment suggests the interactions between the weighting levels for Classes 1 and 3 would have a more significant impact compared to the interactions between the weighting levels for Classes 1 and 2 or Classes 2 and 3, according to Table 13 through Table 15. This intuitively makes sense because a higher percentage of the total assignments come from Classes 1 and 3. Table 14 suggests that varying the weighting levels for Class 2 would have a greater impact on itself; however, varying the weighting levels for Class 2 would cause a less significant change between the other two Classes. Again, this is because Class 2 accounts for a small percentage of the total number of assignments.

The second experiment compares the percentage of recruits actually assigned from each Class versus the total number of assignments. Table 16 through Table 18 displays weighting sensitivity from the 64 scenarios with the weighting levels for Class 1, 2, and 3, respectively:

Table 16. JMP Parameter Estimates for Class 1: Hispanics (Total Assignments)

| **Term** | **Estimate** | **Std Error** | **t Ratio** | **Prob>\|t\|** |
|---|---|---|---|---|
| Intercept | 0.6692824 | 0.009412 | 71.11 | 0.0000* |
| w1 | 0.0447426 | 0.003328 | 13.45 | <.0001* |
| w2 | -0.003472 | 0.003328 | -1.04 | 0.2968 |
| w3 | -0.034848 | 0.003328 | -10.47 | <.0001* |
| (w1-1.5)*(w2-1.5) | -0.003898 | 0.002976 | -1.31 | 0.1905 |
| (w1-1.5)*(w3-1.5) | -0.004752 | 0.002976 | -1.60 | 0.1105 |
| (w2-1.5)*(w3-1.5) | 0.006016 | 0.002976 | 2.02 | 0.0434* |
| (w1-1.5)*(w2-1.5)*(w3-1.5) | 0.0003901 | 0.002662 | 0.15 | 0.8835 |

Table 17. JMP Parameter Estimates for Class 2: African American (Total Assignments)

| **Term** | **Estimate** | **Std Error** | **t Ratio** | **Prob>\|t\|** |
|---|---|---|---|---|
| Intercept | 0.144826 | 0.003734 | 38.79 | <.0001* |
| w1 | -0.003532 | 0.00132 | -2.68 | 0.0075* |
| w2 | 0.0135668 | 0.00132 | 10.28 | <.0001* |
| w3 | -0.004473 | 0.00132 | -3.39 | 0.0007* |
| (w1-1.5)*(w2-1.5) | -0.000764 | 0.001181 | -0.65 | 0.5178 |
| (w1-1.5)*(w3-1.5) | 0.0051834 | 0.001181 | 4.39 | <.0001* |
| (w2-1.5)*(w3-1.5) | -0.001222 | 0.001181 | -1.04 | 0.3008 |
| (w1-1.5)*(w2-1.5)*(w3-1.5) | -0.002091 | 0.001056 | -1.98 | 0.0479* |

Table 18. JMP Parameter Estimates for Class 3: QT Scores (Total Assignments)

| **Term** | **Estimate** | **Std Error** | **t Ratio** | **Prob>\|t\|** |
|---|---|---|---|---|
| Intercept | 0.790429 | 0.006515 | 121.33 | 0.0000* |
| w1 | -0.034607 | 0.002303 | -15.03 | <.0001* |
| w2 | -0.005034 | 0.002303 | -2.19 | 0.0290* |
| w3 | 0.0428685 | 0.002303 | 18.61 | <.0001* |
| (w1-1.5)*(w2-1.5) | 0.0050861 | 0.00206 | 2.47 | 0.0136* |
| (w1-1.5)*(w3-1.5) | 0.0020816 | 0.00206 | 1.01 | 0.3124 |
| (w2-1.5)*(w3-1.5) | -0.004061 | 0.00206 | -1.97 | 0.0489* |
| (w1-1.5)*(w2-1.5)*(w3-1.5) | -0.000545 | 0.001843 | -0.30 | 0.7673 |

The tables above estimate the amount of trade-off that could be expected for an increase in one weighting level for each Class. For example, in Table 16, if the weight for Class 1 increases by one weighting level, the estimate for the percent of assignments from Class 1 would increase approximately 4.47%, Class 2 would decrease approximately 0.35%, and Class 3 would decrease approximately 3.48%. Note this example is solely focusing on Class 1 (i.e., the weighting levels for Classes 2 and 3 are zero, or $w_2 = w_3 = 0$)

The second experiment, like the first experiment, suggests the interactions between the weighting levels for Classes 1 and 3 would have a more significant impact compared to the interactions between the weighting levels for Classes 1 and 2 or Classes 2 and 3, according to Table 16 through Table 18, because a higher percentage of the total assignments come from Classes 1 and 3. Similar to the first experiment, Table 17 suggests that varying the weighting levels for Class 2 would have a greater impact on itself and a lesser impact between the other two Classes because Class 2 accounts for a small percentage of the total number of assignments.

There are limitations for selecting weighting levels to achieve desired outcomes. Due to time constraints, only 30 different randomly generated instances were performed (compared to the 2,000 randomly generated instances for the SL size instances). Also, the experiments were performed based upon the four 369th monthly instances (i.e., ranges listed in Table 5) so there is potential for a large amount of variability compared to the estimated results in Table 13 through

Table 18. Thus, there is no exact answer to achieving desired results based upon the DOEs; however, there are three "Rule of Thumb" scenarios which could achieve desired results:

1) Recall that $0 \leq w_c \leq |C|$. If focusing solely on one Class, it would be expected that maximizing the weight for that Class and minimizing the weight for the other Classes could result in a higher percentage of total assignments for that one Class (e.g., $w_1 = 3$ and $w_2 = w_3 = 0$).
2) Applying a zero weighting for all Classes and establishing a baseline using the statistical results should give a general idea of how the different Classes would be affected with different weighting levels when running the matching algorithm multiple times.
3) The normalized the ratio of the maximum value each Class provides to the total number of assignments should indicate where most of the value comes from. This should impact how many recruits from that respective Class are assigned.

Refer to Figure 27 in Appendix D for further analysis.

**Other Applications**

Although the matching algorithm was developed for the 369th Recruiting Squadron, this matching algorithm could also be applied to other areas of assignment matching problems. The 369th is focused on meeting hard-to-fill jobs and filling jobs with quality recruits while trying to meet percentage goals for each Class. However, Sub-problem 2 is capable of handling a diverse set of objectives. For example, if a company is looking to hire any number of applicants based upon education, experience, skill level, and criminal records, the matching algorithm would associate respective values for each applicant and each job the applicant is qualified for based upon the set criteria. This would output an optimal set of assignments based upon the given objectives and, because this matching algorithm potentially saves the user a significant amount of time over a long period, this would allow the user to focus more time on other areas of need (e.g., setting up and conducting interviews, potentially reducing the amount of man-power needed to create feasible assignments, expediting the assignment process, etc.). There are many different potential problems where this matching algorithm could be applied to produce optimal assignments based upon certain criteria; however, there are limitations as discussed previously, so caution is advised when implementing this algorithm.

## Validation

Four 369th Recruiting Squadron monthly instances were provided for the months of December 2008, February 2009, March 2009, and April 2009, as given in Table 4. Table 19 displays the FY09 goals/targets for the 369th Recruiting Squadron:

Table 19. FY09 Enlisted Accessions Goals/Targets

| | FY09 Goals/Targets |
|---|---|
| Class 1 (Hisp.) | 39.22% |
| Class 2 (Af. Am.) | 11.04% |
| Class 3 (CAT I-IIIA) | 75.40% |

The matching algorithm was run on these four monthly instances. Table 20 and Table 21 displays the matching algorithm results and averages for the four monthly instances on the actual number of assignments for each Class versus total number of assignments and on the actual number of assignments for each Class versus maximum number of assignments for each Class, respectively:

Table 20. Matching Algorithm Results for the Monthly Instances (over total assignments)

| | Dec 2008 | Feb 2009 | Mar 2009 | Apr 2009 | Average |
|---|---|---|---|---|---|
| Class 1 (Hisp.) | 42.50% | 42.99% | 54.05% | 51.04% | 47.65% |
| Class 2 (Af. Am.) | 15.00% | 11.21% | 12.61% | 12.50% | 12.83% |
| Class 3 (CAT I-IIIA) | 75.00% | 89.72% | 93.69% | 89.58% | 87.00% |

Table 21. Matching Algorithm Results for the Monthly Instances (over max # in Class)

| | Dec 2008 | Feb 2009 | Mar 2009 | Apr 2009 | Average |
|---|---|---|---|---|---|
| Class 1 (Hisp.) | 80.95% | 80.70% | 78.95% | 66.22% | 76.71% |
| Class 2 (Af. Am.) | 75.00% | 66.67% | 93.33% | 80.00% | 78.75% |
| Class 3 (CAT I-IIIA) | 100% | 97.96% | 93.69% | 91.49% | 95.79% |

For Class 1 (Hispanics), Class 2 (African Americans), and Class 3 (CAT I-IIIA), the goals/targets are 39.22%, 11.04%, and 75.40% (refer to Table 19). It is evident from Table 20 that, on average, all three goals/targets were exceeded using the matching algorithm provided in this research. For Classes 1, 2, and 3, the matching algorithm exceeded the goals/targets by 8.43, 1.79, and 11.60 percentage points, respectively (or showed an increase of 21.49%, 16.21%, and 15.38%, respectively). The averages were taken from a small sample (i.e., four 369th monthly instances); however, with a larger sample, the matching algorithm provided in this research has the potential to not only meet the user's monthly and yearly goals/targets, but will also save the user several hundred man-hours every year.

# V. Conclusions and Future Work

This section discusses the conclusions and possible future work from this research. First, concluding remarks from this research are discussed. Next, possible work that could be further explored is explained. Finally, the contributions this research provides is stated.

Though the previous research aforementioned is not collectively exhaustive, it enabled this research to find a new method of finding an optimal set of assignments that maximized the total value of those assignments while taking into consideration the different Classes using a weighting structure to articulate which Class (or Classes) to focus on. From the methodology, the matching algorithm was developed and validated. This matching algorithm demonstrated that is has the potential to not only meet the user's goals/targets each month (refer to Table 19 and Table 20), but can also save the user several hundred man-hours every year (refer to Table 9). The extra man-hours could be used in other productive areas such as focusing more hours towards recruiting efforts or working to sign and place recruits to jobs. The results from Table 9 and Table 20 clearly indicate this matching algorithm can be a valuable tool for the 369th Recruiting Squadron.

Due to time constraints, three questions were asked: savings in time, quantity versus quality of assignments, and meeting user goals for percentage of assignments for each Class. All three questions were answered with positive results. However, two questions that could be analyzed involve Sub-problem 2. First, would solving Sub-problem 2 with a multi-objective approach, compared to a single-objective approach, provide better results in terms of percentage of assignments for each Class (i.e., would performing bound analysis to get tighter RHS on the $T_c$ values provide better results in terms of percentage of assignments for each Class)? Second, would solving Sub-problem 2 with a multi-objective approach provide a computational advantage compared to the methods used in this research? Given more time, further analysis could be performed to approximate weighting levels for each Class to achieve desired results. This would require more 369th monthly instances (compared to the four used in this research) in order to better approximate parameter settings for the various categories (as in Table 4 and Table 5). Finally, better methods may exist to increase performance/results of the matching algorithm (e.g., a new VBA algorithm to improve performance).

The contributions for the 369th are evident: a matching algorithm that quickly creates an optimal set of assignments that exceeds the goals/targets set for each Class. However, there are two contributions to the OR community. The first OR contribution involves the complexity of the Main Model Formulation (Model 9). Because this research showed that total unimodularity holds for the Main Problem Formulation, adding additional Sub-problem side constraints does not add to the complexity of the Main Problem Formulation (i.e., the Main Problem Formulation can be solved in polynomial time). The second OR contribution involves the application. Although the matching algorithm was created for the 369th, Sub-problem 2 is capable of handling a diverse set of objectives. The matching algorithm (given the limitations previously defined) could be applied to produce optimal assignments, based upon user criteria, in many different potential problems (e.g., weapons to targets, individuals to jobs, or machines to tasks).

The 369th Recruiting Squadron is considering implementation of the matching algorithm to help optimize the number of assignments and user goals/targets for each month and fiscal years.

## Appendix A. Communication between VBA and LINGO

Appendix A shows how to communicate between VBA and LINGO. Below is an excerpt from the LP model within the LINGO file:

```
Print #fnum, "[soln] max = ";
```

This is followed by the decision variables with coefficients using the LINGO syntax to create the objective function. The "[soln]" is used to capture the objective function value when the LP model is solved using LINGO. After constructing the various constraints for the LP model, below is a method to output to text files for VBA to read as input:

```
Print #fnum, "CALC:"
Print #fnum, "    t0 = @TIME();"
Print #fnum, "    @SET('TERSEO',3);"
Print #fnum, "    @SET('DUALCO',0);"
Print #fnum, "    @SET('STAWIN',0);"
Print #fnum, "    @SOLVE();"
Print #fnum, "    t1 = @TIME();"
Print #fnum, "    t2 = t1 - t0;"
Print #fnum, "    @TEXT('filename1.txt') = t2;"
Print #fnum, "    @TEXT('filename2.txt') = soln;"
Print #fnum, "    @DIVERT('filename3.txt');"
Print #fnum, "    @SOLU(0);"
Print #fnum, "    @DIVERT();"
Print #fnum, "ENDCALC"
Print #fnum,
```

Under the "CALC:" field, the initial time, "t0", is captured. "@SET('TERSEO',3)" sets the level of output to nothing to save overall time. "@SET('DUALCO',0)" forgoes the dual computations (i.e., dual values and ranges), which also saves overall time. "@SET('STAWIN',0)" forgoes the status window display to minimize script command errors sent to LINGO from VBA. "@SOLVE()" solves the given LP model. "t1" captures the time elapsed since solving the LP model and "t2" represents the total time it takes to solve the LP model and stores it in a text file.

Once the LP model is solved, the solution (“[soln]”) is stored in another text file. “@DIVERT('*filename3*.txt')” creates another text file to capture the resulting output from solving the LP model. “@SOLU(0)” is used to capture only the non-zero elements in the resulting output and “@DIVERT()” is used to send the output to the respective text file. “ENDCALC” is used to end the “CALC:” field.

The text files created from solving the LP model can be called from VBA. Below is an excerpt from the VBA code to call the respective text files:

```
Sub Name_of_Subroutine()

   Dim isSolDone As String
   Dim isTimeDone As String
   Dim LINGOapp As Variant

   LINGOapp = Shell("C:\Lingo11\Lingo11 -O" & "LINGOfileName.lg4", 1)

   Sleep WaitLingoOpen
   SendKeys "^u", True

   isSolDone = Dir(FolderDirectory & "filename2.txt")
   isTimeDone = Dir(FolderDirectory & "filename1.txt")
   While isSolDone = ""
      isSolDone = Dir(FolderDirectory & " filename2.txt")
   Wend

   While isTimeDone = ""
      isTimeDone = Dir(FolderDirectory & " filename1.txt")
   Wend

   Sleep WaitLingoClose
   SendKeys "{F10}", True

   Open "filename2.txt" For Input As #1
   Input #1, variable_name1
   Close #1

   If F < 1 Then
      F = 0
   End If
```

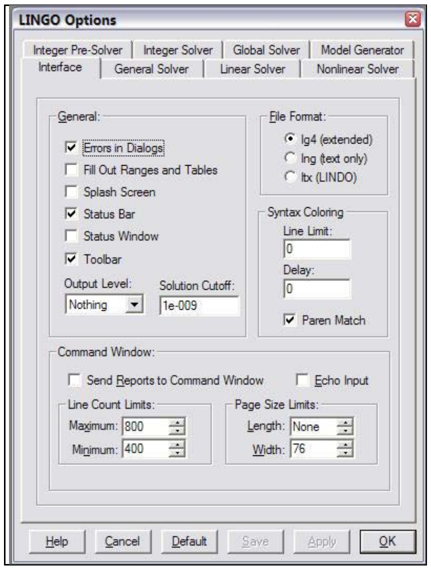
```
        Open " filename1.txt" For Input As #2
        Input #2, variable_name2
        Close #2

    End Sub
```

This VBA algorithm dimensions variable names which are used in the subroutine. It then opens LINGO and the respective LINGO file. A script command is sent to wait a certain period of time ("WaitLingoOpen" is a variable that is user-dimensioned in the VBA code) to allow LINGO to finish opening the file and then another script command is sent to solve the LINGO file. Once the LINGO file is solved and before the LINGO program is closed, the respective text files need to be present. Once it is determined the respective text files are present, a script command is sent to wait a certain period of time ("WaitLingoClose" is another variable that is user-dimensioned in the VBA code) to allow LINGO to finish processing and then another script command is sent to close LINGO. "*variable_name1*" and "*variable_name2*", which are variables that are user-dimensioned in the VBA code, store the values of the respective text files for use in the VBA algorithm. The third text file "*filename3*.txt", which is not referred to in the subroutine above, can be called from VBA for data manipulation however the user feels is necessary. For more detailed information on the VBA code used for the matching algorithm, please contact the person responsible for this thesis.

## Appendix B. Additional Figures and Tables

Appendix B contains additional figures and tables discussed in the research. First, Figure 2 and Figure 3 shows the changes to the LINGO default settings. Next, Figure 4 through Figure 24 shows the different distributions for three group's times and the quantity versus quality of assignments. Finally, Table 22 through Table 25 shows the paired t-tests for the quantity versus quality of assignments.


LINGO Options
Integer Pre-Solver | Integer Solver | Global Solver | Model Generator
Interface | General Solver | Linear Solver | Nonlinear Solver
Help | Cancel | Default | Save | Apply | OK

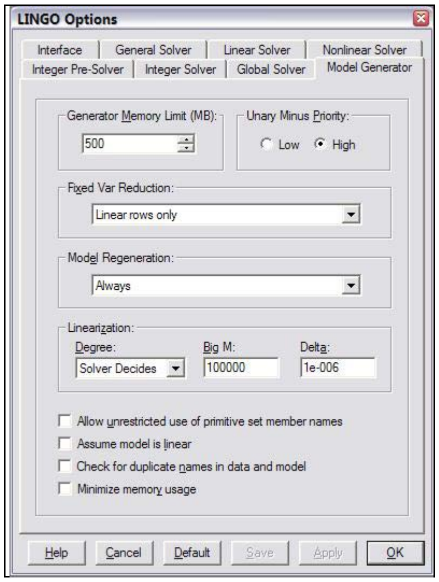
General:
[✓] Errors in Dialogs
[ ] Fill Out Ranges and Tables
[ ] Splash Screen
[✓] Status Bar
[ ] Status Window
[✓] Toolbar
Output Level: Nothing
Solution Cutoff: 1e-009

File Format:
[● lg4 (extended)] lng (text only) ltx (LINDO)

Syntax Coloring
Line Limit: 0
Delay: 0
[✓] Paren Match

Command Window:
[ ] Send Reports to Command Window
[ ] Echo Input
Line Count Limits: Maximum: 800 Minimum: 400
Page Size Limits: Length: None Width: 76

Figure 2. LINGO Settings under Interface Tab


LINGO Options
Interface | General Solver | Linear Solver | Nonlinear Solver
Integer Pre-Solver | Integer Solver | Global Solver | Model Generator
Help | Cancel | Default | Save | Apply | OK

Generator Memory Limit (MB): 500

Unary Minus Priority: Low [● High]

Fixed Var Reduction: Linear rows only

Model Regeneration: Always

Linearization:
Degree: Solver Decides
Big M: 100000
Delta: 1e-006

[ ] Allow unrestricted use of primitive set member names
[ ] Assume model is linear
[ ] Check for duplicate names in data and model
[ ] Minimize memory usage

Figure 3. LINGO Settings under Model Generator Tab

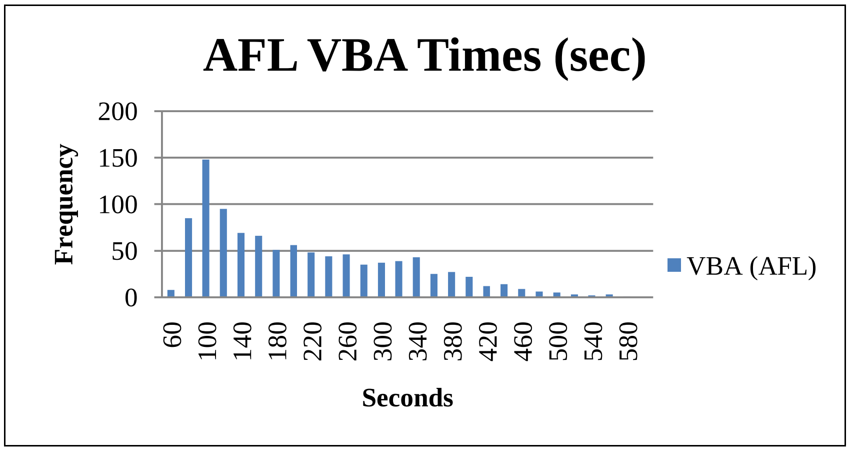


Figure 4. AFL VBA Times Histogram

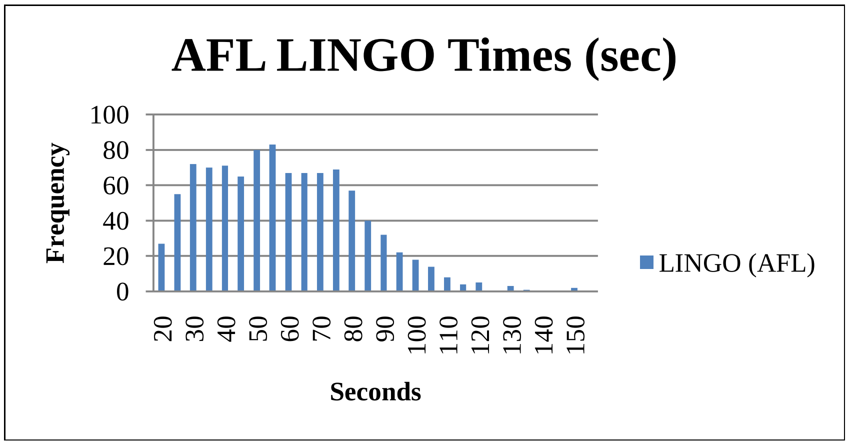


Figure 5. AFL LINGO Times Histogram

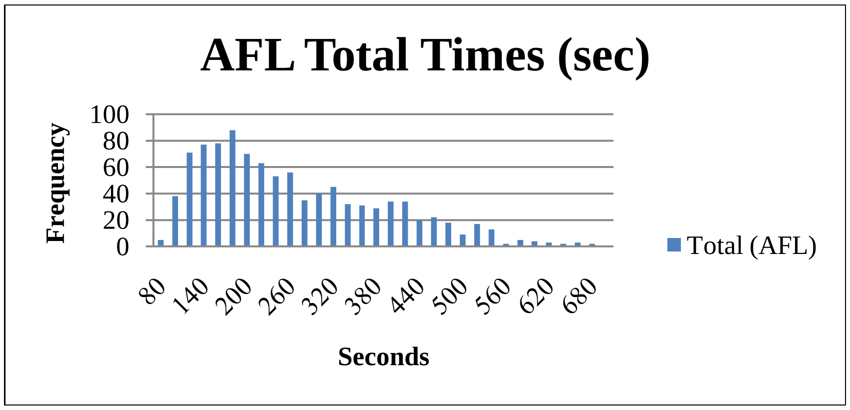


Figure 6. AFL Total Times Histogram

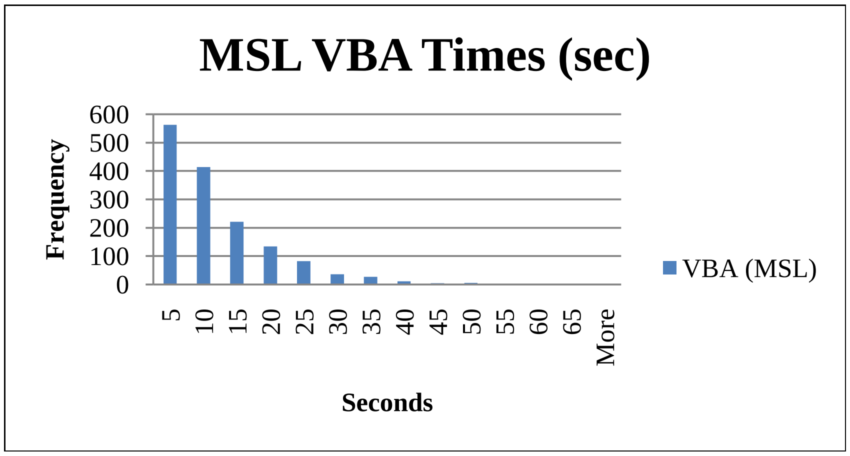


Figure 7. MSL VBA Times Histogram

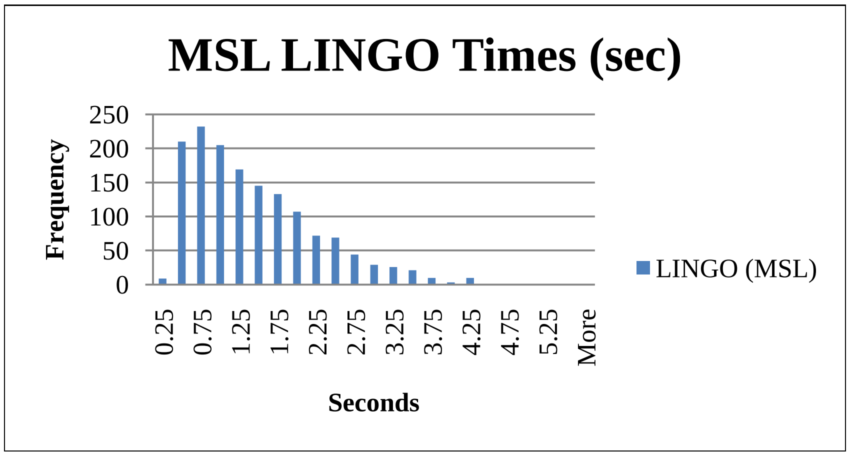


Figure 8. MSL LINGO Times Histogram

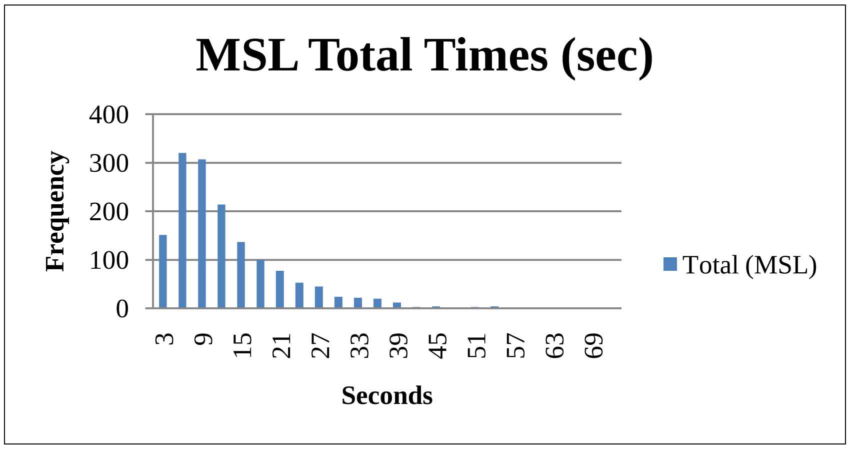


Figure 9. MSL Total Times Histogram

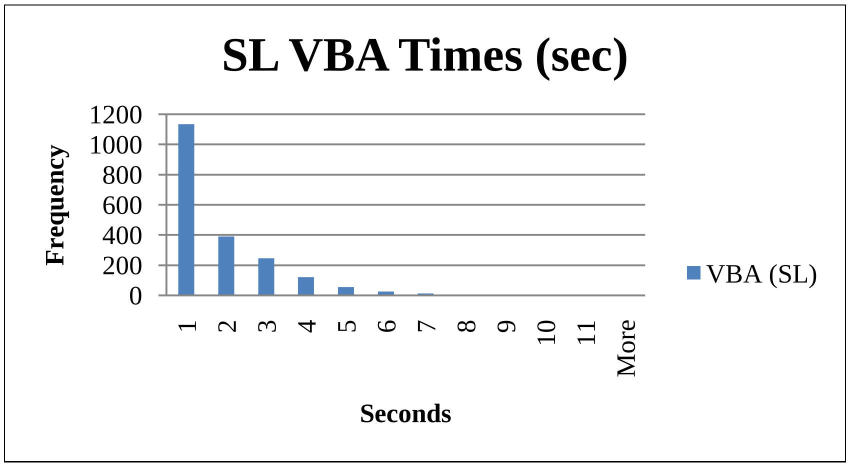


Figure 10. SL VBA Times Histogram

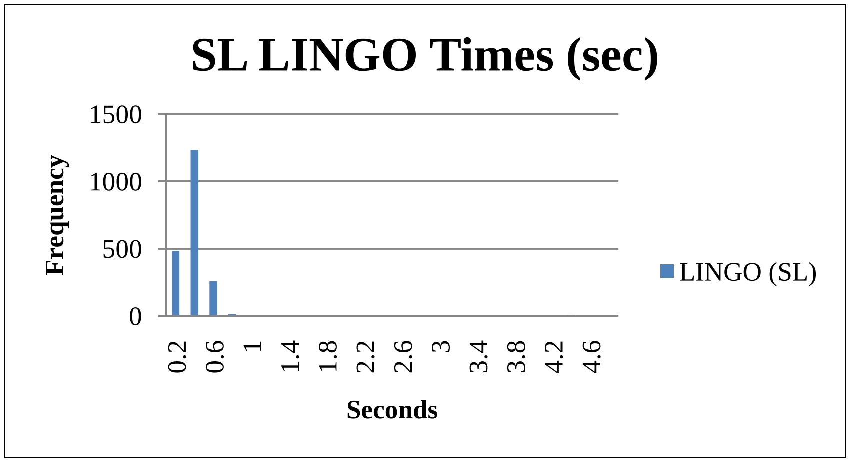


Figure 11. SL LINGO Times Histogram

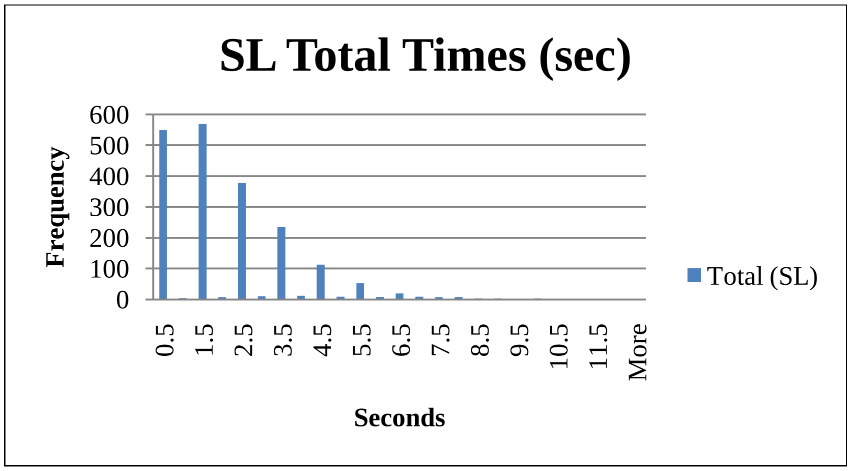


Figure 12. SL Total Times Histogram

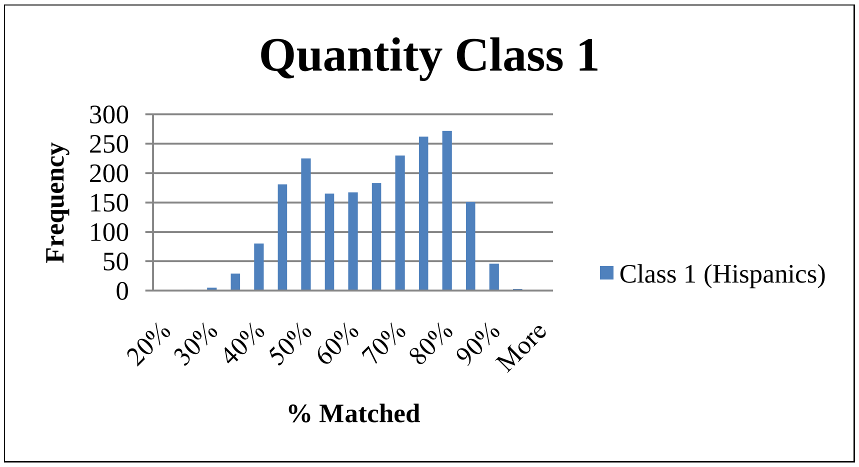


Figure 13. Quantity of Matches (Class 1: Hispanics)

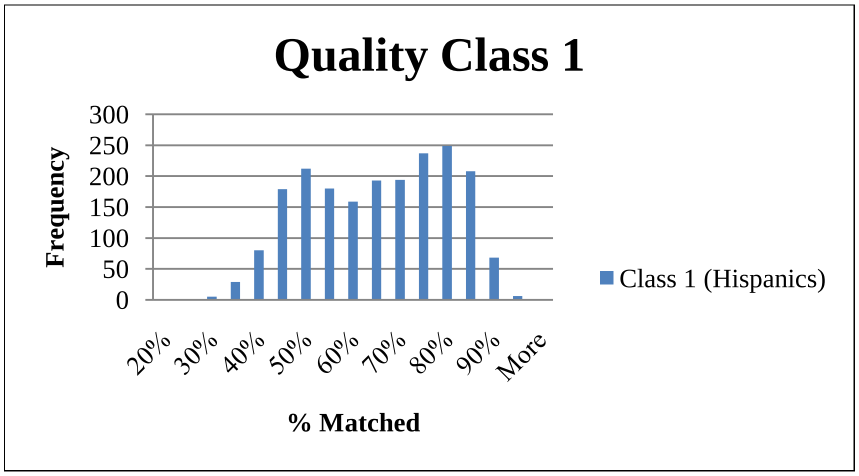


Figure 14. Quality of Matches (Class 1: Hispanics)

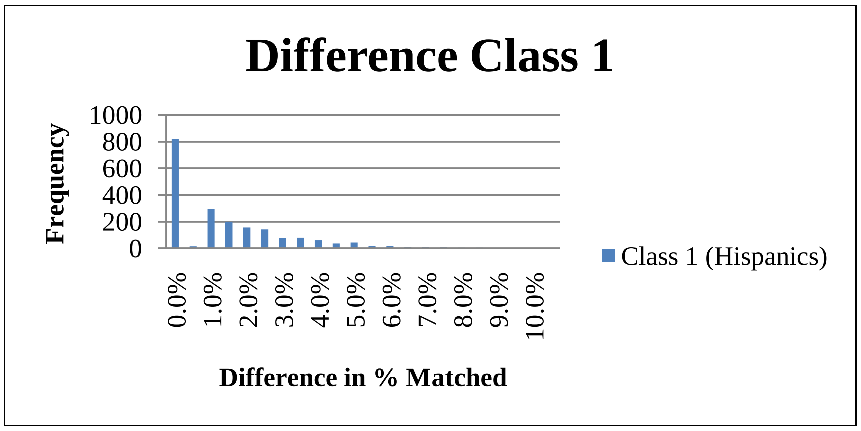


Figure 15. Difference of Matches (Class 1: Hispanics)

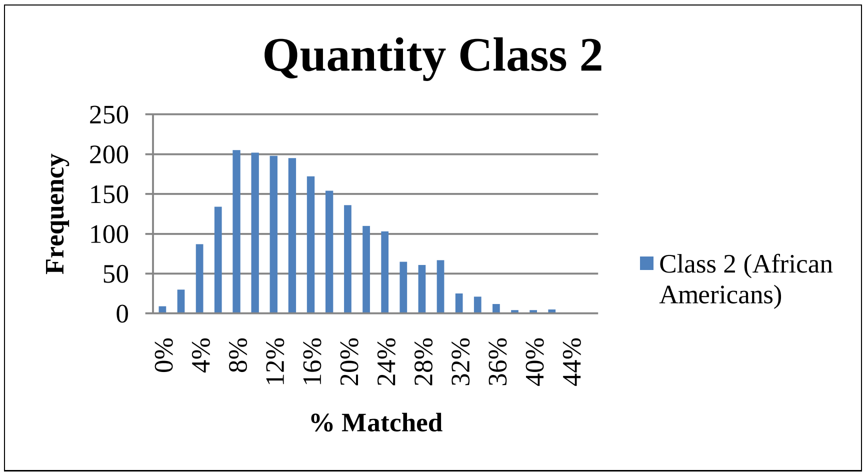


Figure 16. Quantity of Matches (Class 2: African Americans)

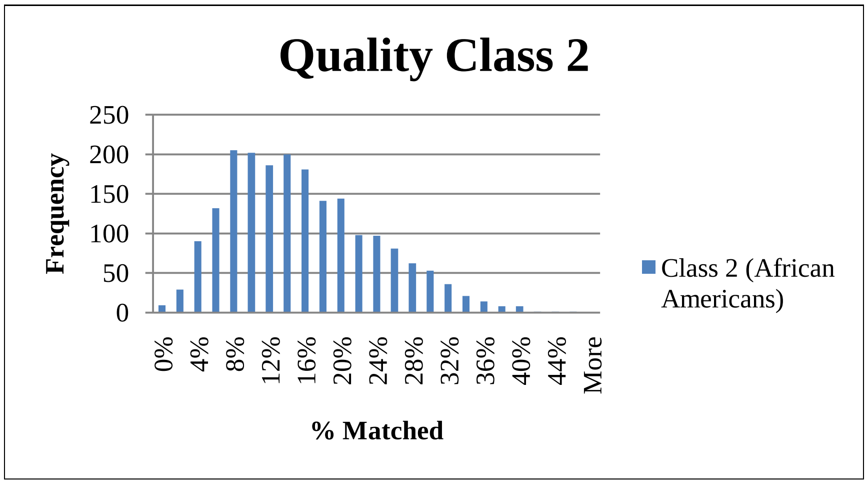


Figure 17. Quality of Matches (Class 2: African Americans)

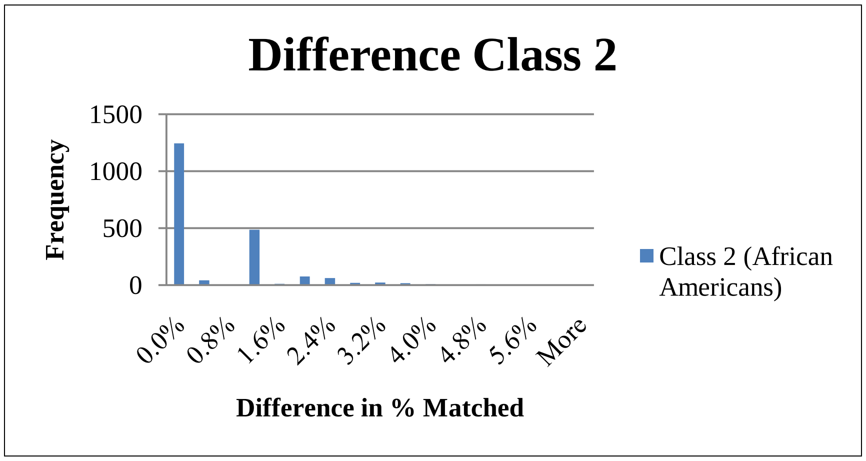


Figure 18. Difference of Matches (Class 2: African Americans)

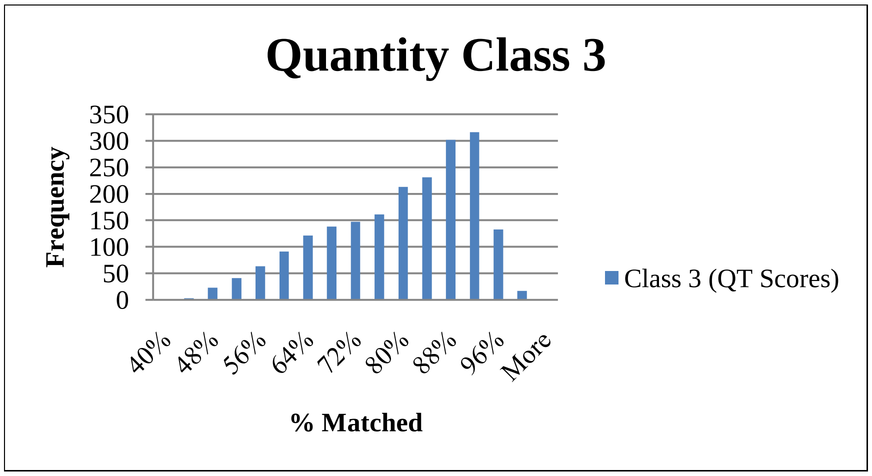


Figure 19. Quantity of Matches (Class 3: QT Scores)

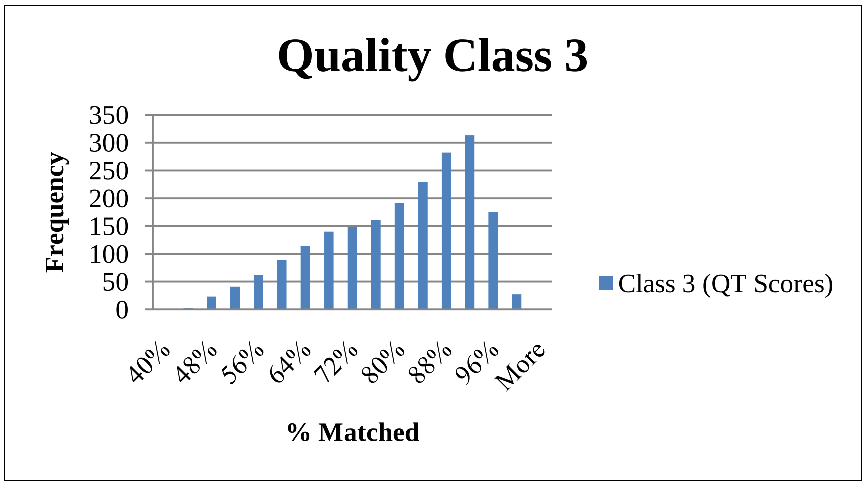


Figure 20. Quality of Matches (Class 3: QT Scores)

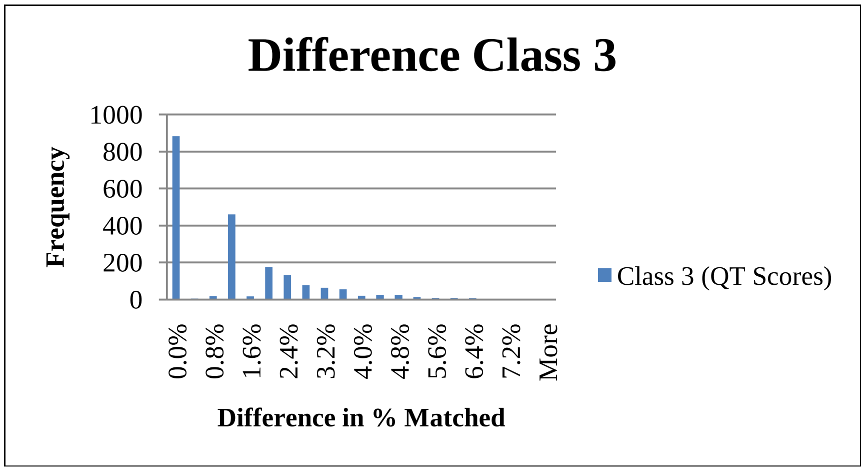


Figure 21. Difference of Matches (Class 3: QT Scores)

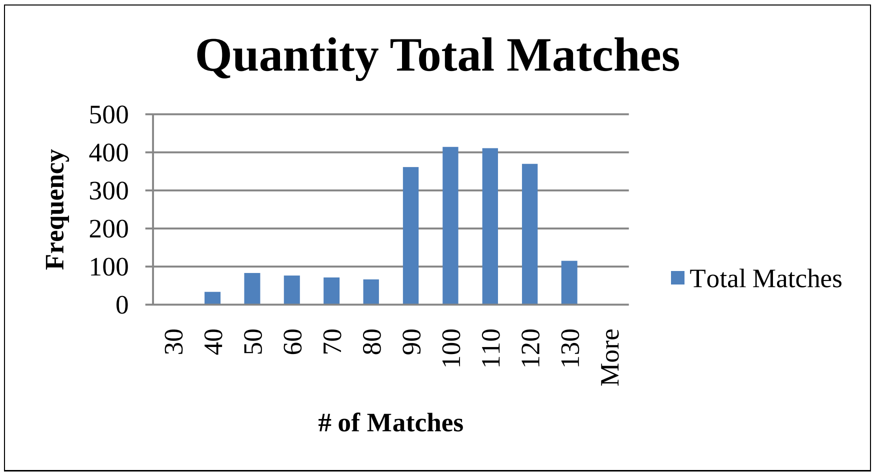


Figure 22. Quantity of Matches (Total # of Matches)

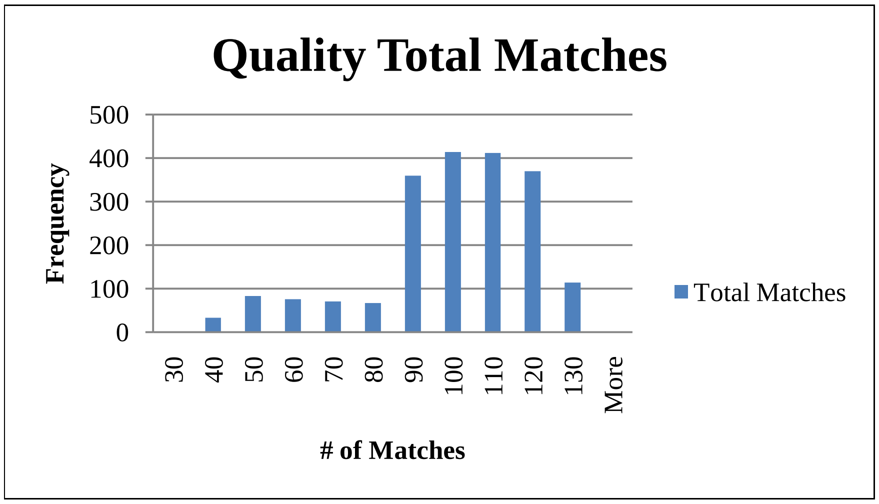


Figure 23. Quality of Matches (Total # of Matches)

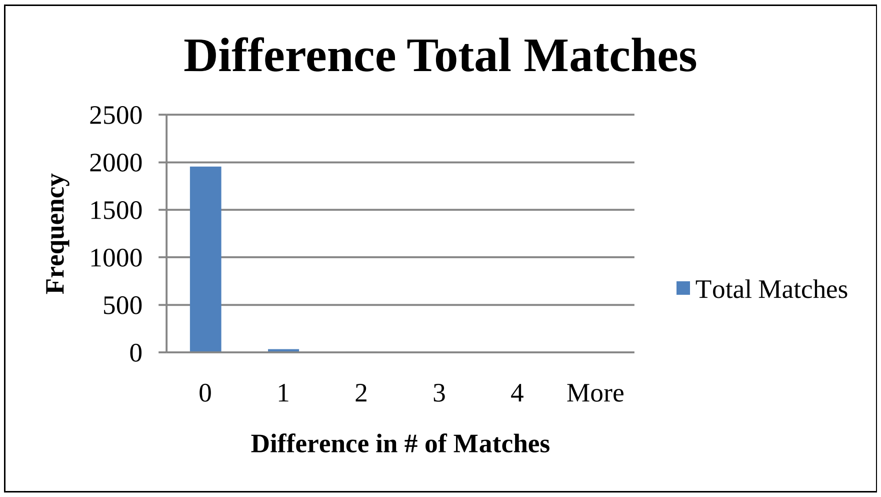


Figure 24. Difference of Matches (Total # of Matches)

Table 22. Paired t-Test for Class 1 (Hispanics)

| t-Test: Paired Two Sample for Means | Class 1 (Hispanics) | |
|---|---|---|
| | *Quantity* | *Quality* |
| Mean | 0.623637617 | 0.629460802 |
| Variance | 0.020271316 | 0.021867806 |
| Observations | 2000 | 2000 |
| Pearson Correlation | 0.990890815 | |
| Hypothesized Mean Difference | 0 | |
| df | 1999 | |
| t Stat | -12.80159797 | |
| P(T<=t) one-tail | 2.02018E-36 | |
| t Critical one-tail | 1.645616249 | |
| P(T<=t) two-tail | 4.04036E-36 | |
| t Critical two-tail | 1.96115137 | |

Table 23. Paired t-Test for Class 2 (African Americans)

| t-Test: Paired Two Sample for Means | Class 2 (African Americans) | |
|---|---|---|
| | *Quantity* | *Quality* |
| Mean | 0.145669631 | 0.146894085 |
| Variance | 0.006240726 | 0.00648056 |
| Observations | 2000 | 2000 |
| Pearson Correlation | 0.992561337 | |
| Hypothesized Mean Difference | 0 | |
| df | 1999 | |
| t Stat | -5.56358808 | |
| P(T<=t) one-tail | 1.49914E-08 | |
| t Critical one-tail | 1.645616249 | |
| P(T<=t) two-tail | 2.99828E-08 | |
| t Critical two-tail | 1.96115137 | |

Table 24. Paired t-Test for Class 3 (QT Scores)

| t-Test: Paired Two Sample for Means | Class 3 (QT Scores) | |
|---|---|---|
| | *Quantity* | *Quality* |
| Mean | 0.775085445 | 0.778867135 |
| Variance | 0.015069818 | 0.015632544 |
| Observations | 2000 | 2000 |
| Pearson Correlation | 0.990785052 | |
| Hypothesized Mean Difference | 0 | |
| df | 1999 | |
| t Stat | -9.965099988 | |
| P(T<=t) one-tail | 3.66195E-23 | |
| t Critical one-tail | 1.645616249 | |
| P(T<=t) two-tail | 7.3239E-23 | |
| t Critical two-tail | 1.96115137 | |

Table 25. Paired t-Test for Total # of Matches

| t-Test: Paired Two Sample for Means | Total Matches | |
|---|---|---|
| | *Quantity* | *Quality* |
| Mean | 94.8935 | 94.8655 |
| Variance | 419.4088622 | 419.5321758 |
| Observations | 2000 | 2000 |
| Pearson Correlation | 0.999948472 | |
| Hypothesized Mean Difference | 0 | |
| df | 1999 | |
| t Stat | 6.022019327 | |
| P(T<=t) one-tail | 1.0223E-09 | |
| t Critical one-tail | 1.645616249 | |
| P(T<=t) two-tail | 2.0446E-09 | |
| t Critical two-tail | 1.96115137 | |

## Appendix C. VBA and LINGO Completion Times

Appendix C shows the VBA and LINGO completion times. Table 26 displays the VBA completion times to create the LP models into LINGO files using a PC with Windows 7 Ultimate (64-bit) as an operating system with an Intel Core 2 Duo E8500 (3.16 GHz), and 4.00 GB of RAM:

Table 26. VBA Completion Times using a *n* x *n* Recruit/Job Matrix

| # of Elements in the Matrix | *n* x *n* Matrix | VBA Completion Times (sec) |
|---|---|---|
| 10000 | 100 | 0.30 |
| 22500 | 150 | 0.69 |
| 40000 | 200 | 1.22 |
| 62500 | 250 | 1.95 |
| 90000 | 300 | 2.84 |
| 122500 | 350 | 3.84 |
| 160000 | 400 | 5.08 |
| 202500 | 450 | 6.52 |
| 250000 | 500 | 7.98 |
| 302500 | 550 | 9.73 |
| 360000 | 600 | 11.56 |
| 422500 | 650 | 13.78 |
| 490000 | 700 | 15.73 |
| 562500 | 750 | 18.28 |
| 640000 | 800 | 20.89 |
| 722500 | 850 | 23.80 |
| 810000 | 900 | 26.40 |
| 902500 | 950 | 29.33 |
| 1000000 | 1000 | 32.22 |
| 1210000 | 1100 | 39.78 |
| 1440000 | 1200 | 47.05 |
| 1690000 | 1300 | 55.69 |
| 1960000 | 1400 | 64.98 |
| 2402500 | 1550 | 80.47 |
| 2890000 | 1700 | 97.09 |
| 3422500 | 1850 | 117.25 |

The 1850x1850 matrix was the limit reached when Excel needed to be reset in order to perform the run. Table 27 displays the *n* x *n* recruit/job parameter settings for the time runs:

Table 27. *n* x *n* Recruit/Job Matrix Parameter Settings

| Parameters | Settings |
|---|---|
| # Recruits | *n* |
| # Jobs | *n* |
| % Hispanic | 40 |
| % African Amer. | 50 |
| % QT Score $\geq$ 50 | 60 |
| # Hard-to-fill Jobs | ~(*n*/2) |
| % Availability | 75 |
| % Preferred | 60 |

These are arbitrary settings used to create a denser recruit/job matrix compared to the randomly generated instances based upon the four 369th monthly instances. Using these settings made it more difficult for VBA to create the LP models into LINGO files so slower times were expected. Figure 25 shows the graphical representation of the VBA computational times to completion (in seconds):

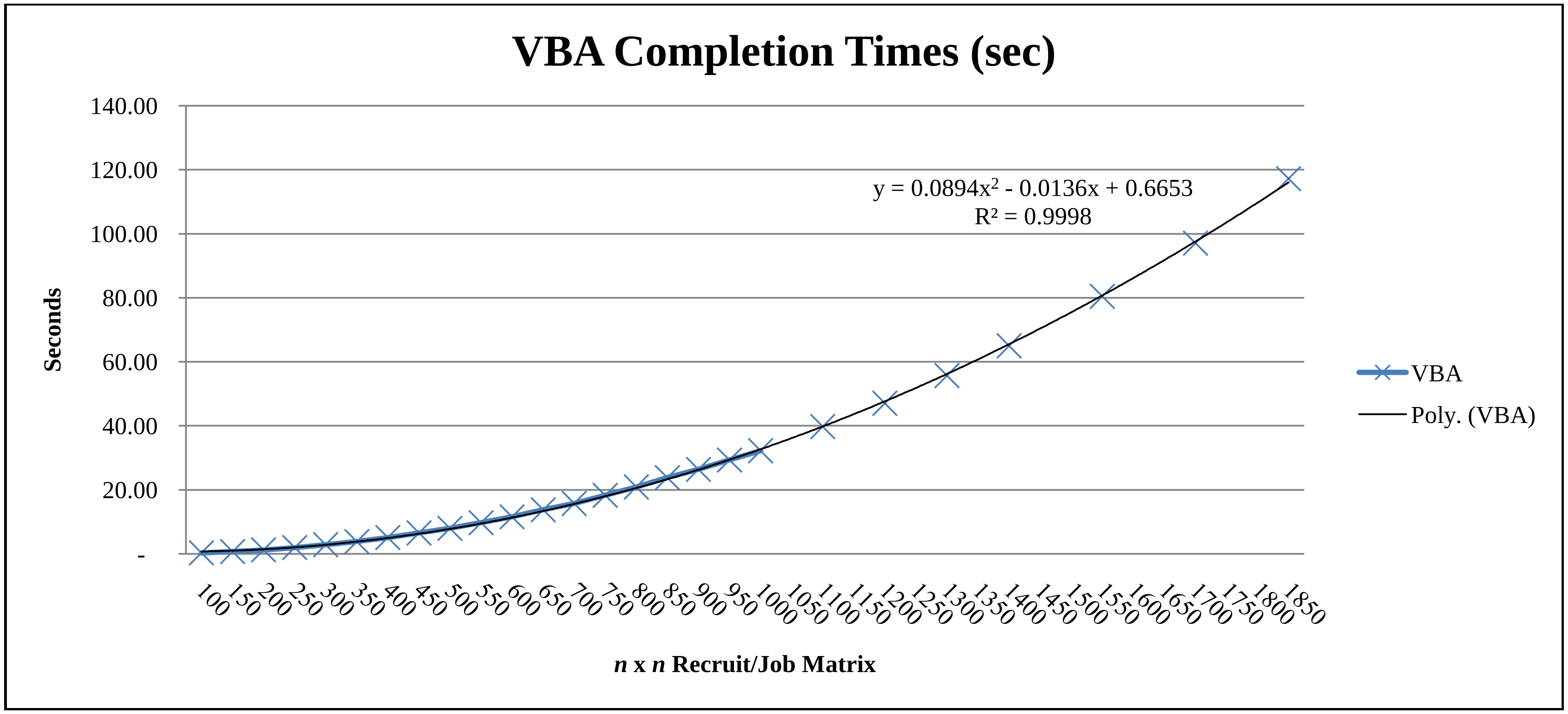


Figure 25. VBA Completion Times

A trendline, with the respective line equation and R-squared value, was added using Excel's trendline function. It is expected that the times for VBA to create larger LP models into LINGO files would take a polynomial amount of time to solve. Table 28 displays the LINGO computational times (in seconds) to solve the respective LP models:

Table 28. LINGO Completion Times using the *n* x *n* Recruit/Job Matrix

| # of Elements in the Matrix | *n* x *n* Matrix | LINGO Completion Times (sec) |
|---|---|---|
| 10000 | 100 | 3.00 |
| 62500 | 250 | 13.00 |
| 302500 | 500 | 113.00 |
| 562500 | 750 | 465.00 |
| 1000000 | 1000 | 1379.00 |

Figure 26 shows the graphical representation of the LINGO times to solve the respective *n* x *n* recruit/job matrices (in seconds):

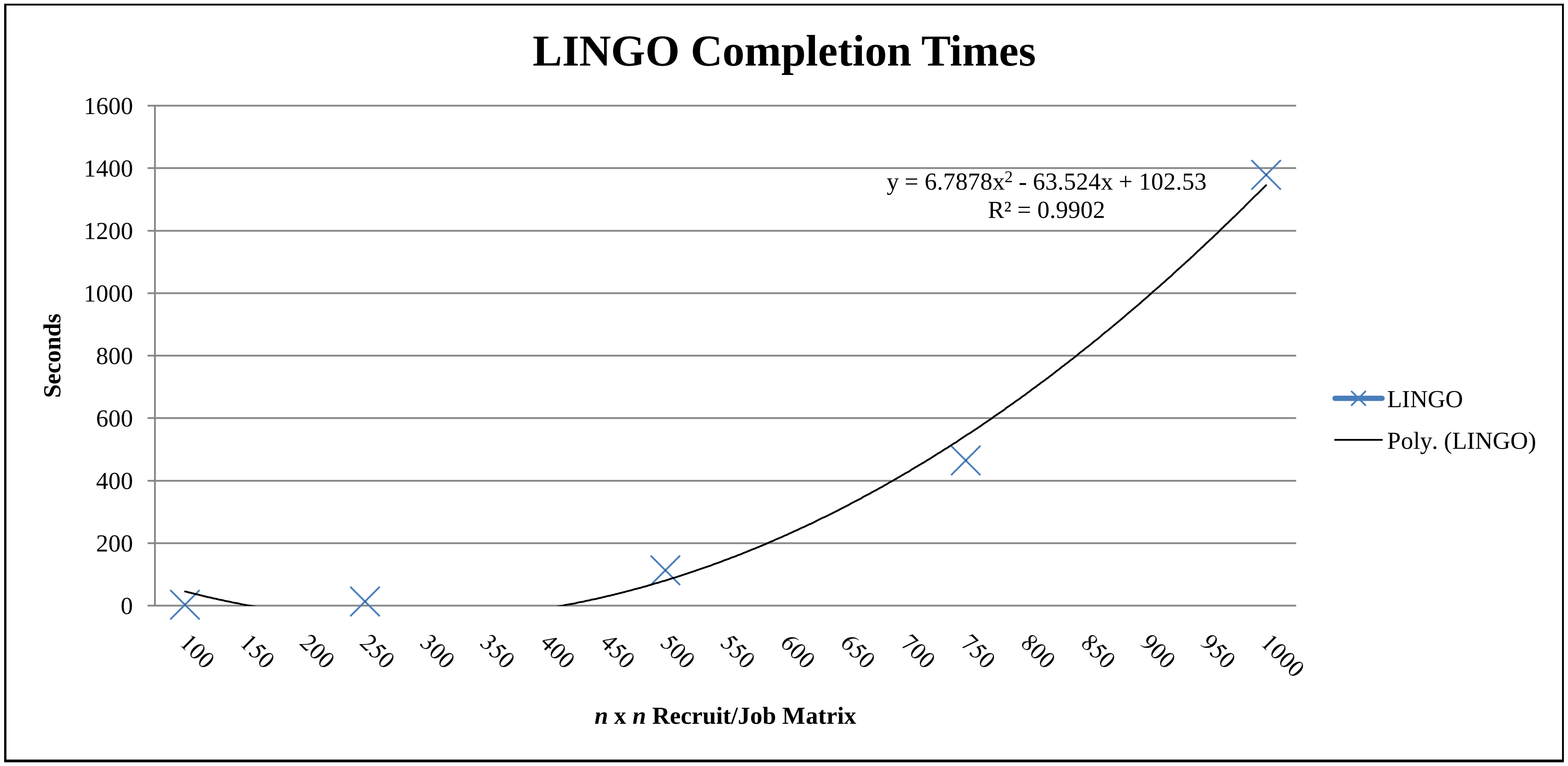
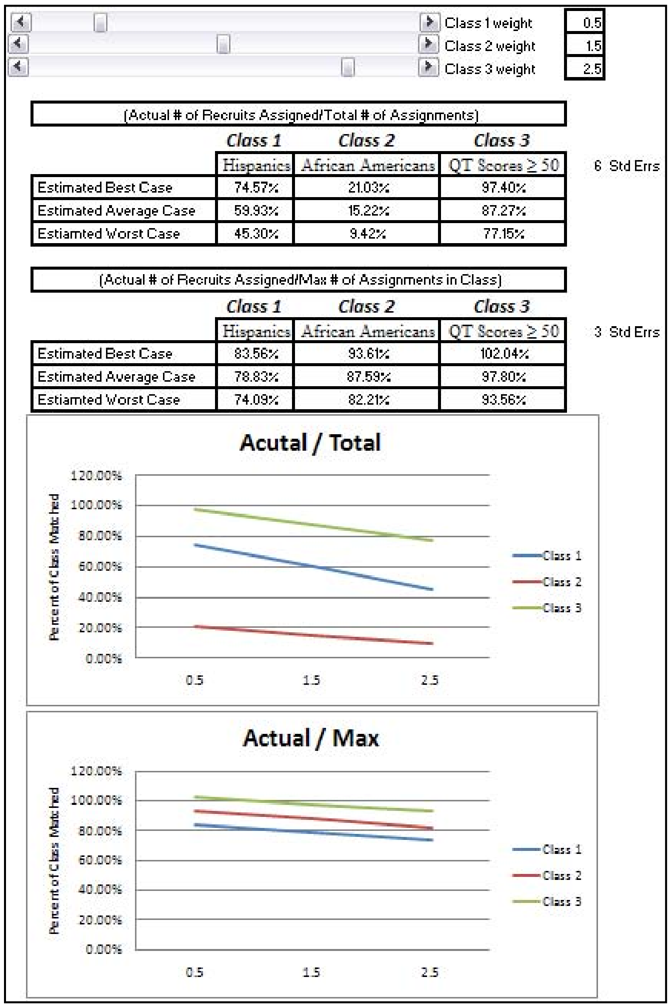


Figure 26. LINGO Completion Times

Again, using Excel's trendline function, the respective line equation and R-squared value are shown. Due to time constraints, only a limited number of instances (from Table 26) were collected. Although the R-squared value is high, it is uncertain how long it would take to solve larger instances of the *n* x *n* matrices using the parameters in Table 27.

## Appendix D. Further Weighting Sensitivity Analysis for Different Classes

Appendix D discusses further weighting sensitivity analysis for the different Classes.



| | |
|---|---|
| Class 1 weight | 0.5 |
| Class 2 weight | 1.5 |
| Class 3 weight | 2.5 |

(Actual # of Recruits Assigned/Total # of Assignments)

| | Class 1 | Class 2 | Class 3 | |
|---|---|---|---|---|
| | Hispanics | African Americans | QT Scores ≥ 50 | 6 Std Errs |
| Estimated Best Case | 74.57% | 21.03% | 97.40% | |
| Estimated Average Case | 59.93% | 15.22% | 87.27% | |
| Estiamted Worst Case | 45.30% | 9.42% | 77.15% | |

(Actual # of Recruits Assigned/Max # of Assignments in Class)

| | Class 1 | Class 2 | Class 3 | |
|---|---|---|---|---|
| | Hispanics | African Americans | QT Scores ≥ 50 | 3 Std Errs |
| Estimated Best Case | 83.56% | 93.61% | 102.04% | |
| Estimated Average Case | 78.83% | 87.59% | 97.80% | |
| Estiamted Worst Case | 74.09% | 82.21% | 93.56% | |

Figure 27. Weighting Scale based upon the DOEs using JMP

Figure 27 shows a weighting scale (with the associated number of standard deviations) created in Excel using the parameter estimates based upon the DOEs using JMP (Table 13 through Table 18). This allows the user to examine, in real time, how weighting each Class might affect the results in all Classes from both experiments.

However, these parameter estimates are based upon the four 369th monthly instances (i.e., ranges listed in Table 5), so caution is advised when using the weighting scale.

The first "Rule of Thumb" scenario is self explanatory. It would be expected that weighting one Class much higher than the other Classes could result in a higher percentage of total assignments for that Class (e.g., $w_1 = 3$ and $w_2 = w_3 = 0$). The second scenario could meet the user's goal in percentage of assignments from each Class by applying a zero weighting for all Classes (i.e., $w_1 = w_2 = w_3 = 0$), which would also establish a baseline. Using the baseline statistical results should give a general idea of how the different Classes would be affected with different weighting levels when the matching algorithm is run multiple times. The third scenario uses the normalized ratio of the maximum value each Class provides to the total number of assignments. Since each recruit has some value for some job (i.e., $v_{r,j}$), each Class will contribute a certain amount of value to the total number of assignments created. This allows the user to get an idea of how weighting one Class affects the other Classes. For example, suppose 34%, 7%, and 59% of Classes 1, 2, and 3, respectively, make up the amount of total value that comes from each Class. From these percentages, it would be expected that a higher weighting for Class 1 would have a minimal impact on Class 2 compared to Class 3, a higher weighting for Class 2 would have a minimal impact on the other two Classes, and a higher weighting for Class 3 would have a minimal impact on Class 2 compared to Class 1. Although there is no exact percentage difference for each Class, using the amount of value that comes from each Class should help determine roughly how the weighting levels should be set.

## Appendix E. Main Problem Formulation Constraint Matrix is Totally Unimodular

Appendix E will show that the constraint matrix for the Main Problem Formulation (Model 9) is totally unimodular; however, it is important to show that total unimodularity holds for any assignment matching problem. Recall that an assignment matching problem is viewed as a bipartite graph with a set of vertices and edges, $G = (V_1, V_2, E)$, where the set of vertices are disjoint such that $V = (V_1 \cup V_2)$ and every edge $E$ has one endpoint in $V_1$ and the other endpoint in $V_2$ (Wolsey 1998). Let $A$ be the constraint matrix of $G$. It is well known that if $G$ is bipartite, then $A$ is totally unimodular (see Gass (1985), Wolsey (1998), Burkard and Cela (1999), Vempala (2003), and IMADA (2009)). Vempala (2003) shows this result by induction on $k$ which represents any $k$ x $k$ sub-matrix $A' \subseteq A$. This is true for $k = 1$ because each element in $A$ is 0 or ±1 by construction. Now assume this result is true for every ($k$-1) x ($k$-1) sub-matrix of $A$. There are three cases to consider (Vempala 2003):

1) If $A'$ has a row of 0's, then the det($A'$) is 0.
2) If $A'$ has a row with a single ±1, then it is possible to expand around the ±1 and use the induction hypothesis on the ($k$-1) x ($k$-1) sub-matrix to figure out that the det($A'$) is 0 or ±1.
3) If each row of $A'$ has more than one ±1, then $A'$ comes from the upper portion of the constraint matrix because the lower half is the identity matrix. So the rows of $A'$ can be split into two sections corresponding to the partitioning of the vertices of $G$ (since $G$ is bipartite). However, the same vectors are obtained by summing the rows from each section because each edge from $E$ is incident to only one vertex from each section. This means that each column

has a 1 in each section which implies that *A*' is linearly dependent (i.e., det(*A*') is 0).

With the resulting proof that the constraint matrix of bipartite graphs are totally unimodular and with Gass's (1985) theorem that if the RHS values are integer then all the basic feasible solutions have integer values, it is clear that the models for Sub-problem 1 (Model 7) and Sub-problem 2 (Model 8) result in integer solutions because both Sub-problems are assignment matching problems which are viewed as bipartite graphs.

The Main Problem Formulation (Model 9) is also an assignment matching problem which is viewed as a bipartite graph, but also includes additional Sub-problem side constraints. Therefore, it is necessary to show that the Sub-problem side constraints do not affect the total unimodularity of the constraint matrix, which is represented by *P*, in Model 9. Let *P*' ⊆ *P* represent constraints (28) and (29) in Model 9. If it can be shown that the RHS values of *P* are integer and the Sub-problem side constraints of *P* are linear combinations of *P*', then *P* is totally unimodular:

1) The RHS values for constraints (28) and (29) in Model 9 are 1. The values for *F* in Sub-problem 1 and $T_c$ in Sub-problem 2, $\forall c \in C$, are integer by using the result from above. Also, for Sub-problem 2, it is easy to see that since $T_c$ is integer $\forall c \in C$, which implies $T_c - \sum_{r \in R} \sum_{j \in J} \mathrm{Rec}_{r,c} D_{r,j} E_{r,j}$ is integer $\forall c \in C$, then the optimal value of $s_c$ is also integer $\forall c \in C$. Since the optimal value of $s_c$ is integer $\forall c \in C$, then $T_c - s_c$ is also some integer $\forall c \in C$. Note that $s_c$ is not restricted to be integer in Model 9, but finding an optimal solution

will result in an integer value for $s_c$, $\forall c \in C$. Therefore, the RHS values for *P* are integer.

2) Recall the Sub-problem side constraints in Model 9:

$$\sum_{r \in R} \sum_{j \in H} D_{r,j} E_{r,j} \geq F \qquad \text{(Sub-problem 1)} \qquad (30)$$

$$\sum_{r \in R} \sum_{j \in J} \text{Rec}_{r,c} D_{r,j} E_{r,j} + s_c \geq T_c \qquad \forall\ c \in C \text{ (Sub-problem 2).} \qquad (31)$$

Also, recall constraints (28) and (29) in Model 9:

$$\sum_{r \in R} D_{r,j} E_{r,j} \leq 1 \qquad \forall\ j \in J \qquad (28)$$

$$\sum_{j \in J} D_{r,j} E_{r,j} \leq 1 \qquad \forall\ r \in R. \qquad (29)$$

Let *Y* and *U* represent the constraint matrices for the Sub-problem 1 and 2 side constraints, respectively, where *Y* and *U* ⊆ *P*. It is evident that a combination of constraints (28) and (29) will form *Y* because both are using the same sets *R* and *J*, where constraint (28) is summed over set *R* and constraint (29) is summed over set *J* and *Y* is summed over both sets *R* and *J* (recall that *H* ⊆ *J*). Also, it is evident that a combination of constraints (28) and (29) will form *U*, using the same reasoning as above, where *U* is summed over both sets *R* and *J* based upon the $\text{Rec}_{r,c}$, $\forall c \in C$. So, it is clear that *Y* and *U* are linear combinations of constraints (28) and (29) in Model 9.

Because the RHS values for *P* are integer and because the Sub-problem side constraints are linear combinations of *P'*, *P* remains totally unimodular and Model 9 can be solved as a LP (i.e., it can be solved in polynomial time).

## Appendix F. Blue Dart

### Assignment Efficiency in the Modern Era

In May 2009, the Secretary of the Air Force Public Affairs stated that the Air Force will spend over $640 million in fiscal year 2010 for recruiting and retaining critical wartime skills such as explosive ordnance disposal (EOD), medical, intelligence, contracting, and special operations (Lyle 2009). Additionally, *The Washington Post* (October 2009) stated that with the current state of the economy and the percentage of unemployment increasing, military recruiters have been able to meet their objectives in both numbers and quality recruits for all components of active duty and reserve forces (Tyson 2009). This increased availability of quality recruits naturally suggests that available jobs can be filled by high quality individuals, which gives the armed forces more flexibility in recruiting and allows the Services to enforce high entry standards.

With the flood of potential recruits, it is no easy task to simply assign a recruit to a job. For example, suppose there are 25 recruits and 25 available jobs in a month. The number of different ways to assign these recruits to the available jobs exceeds the number of known stars in the universe (i.e., $7 \times 10^{22}$), according to a 2003 CNN news source. However, in a typical month for a recruiting squadron, there are over 100 recruits and over 100 available jobs, which mean the number of different ways to assign these recruits to the available jobs is indeed astronomical.

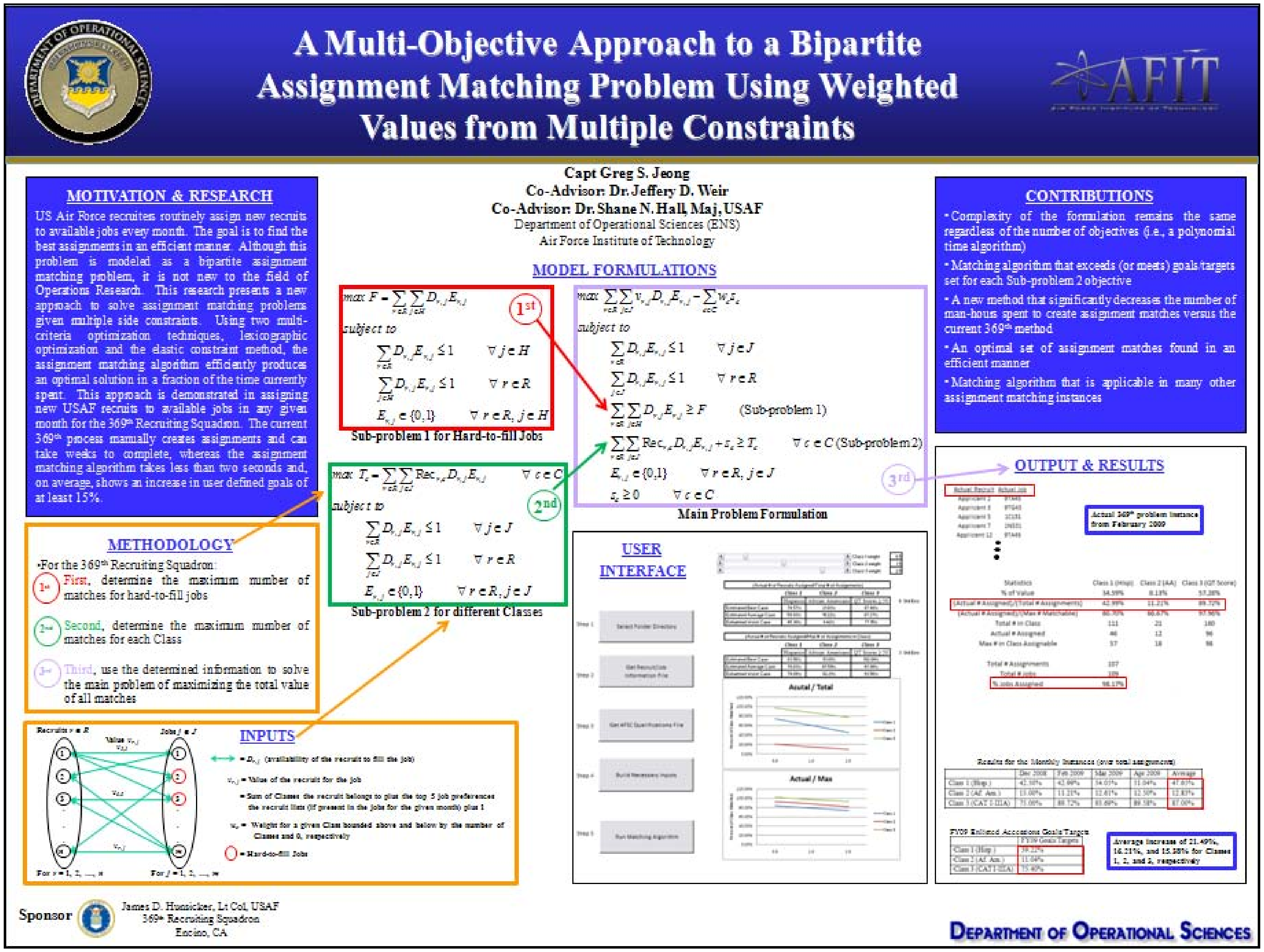
US Air Force recruiters face this “astronomical” task of assigning new recruits to available jobs every month with the goal of creating the best possible assignments for the Air Force and for the individual recruit, and they need to complete this task quickly. The current process to create assignments can take several days and even weeks to complete, as in the case for the 369th Recruiting Squadron.  For example, if it takes 5 hours every day for 5 days to create the best possible assignments each month, then this task would require 25 hours each month, and, over a one year period, would require 300 hours to create the best possible assignments.  If there is a way to do this more quickly, the time saved could be used for other important tasks such as increased recruiting efforts and more one-on-one interaction with potential recruits.

Researchers at the Air Force Institute of Technology (AFIT) used math to model the assignment process.  As part of this research, an assignment matching tool was developed that creates the best possible assignments in a *fraction* of the time currently spent.  With a click of a few buttons, what would have taken dozens of hours each month, now only takes a few seconds.  The researchers demonstrated and validated this tool using four previous months of data provided by the 369th Recruiting Squadron, and, on average, creating assignments using the assignment matching tool took less than two seconds.

Creating assignments more quickly is important; however, there are many goals recruiters would like to meet with these assignments.  Each year, goals are established to enlist recruits that meet certain needs of the Air Force, and trying to create the best possible assignments that meet the yearly goals (let alone *exceed* the yearly goals) only makes the assignment process more difficult.  Often times, recruiters fall short of these goals due to the difficulty in creating assignments in a given time period.  However, the assignment matching tool developed at AFIT is not only capable of quickly creating assignments, but it is capable of creating *the best possible* assignments given the recruiter's goals.  The four previous months of data were used to demonstrate this capability and, on average, the assignment matching tool created the best possible assignments with an increase in the recruiter’s goals of at least 15%.

Even with the increased availability of quality recruits, this assignment matching tool shows very promising results for which recruiters should be very optimistic. The assignment matching tool has demonstrated the capability to not only help recruiters quickly assign new recruits to available jobs, but also to help recruiters create possible assignments for any given month that best meet the needs of the Air Force and the needs of the individual recruit.

Appendix G.  Summary Chart



# A Multi-Objective Approach to a Bipartite Assignment Matching Problem Using Weighted Values from Multiple Constraints

Capt Greg S. Jeong
Co-Advisor: Dr. Jeffery D. Weir
Co-Advisor: Dr. Shane N. Hall, Maj, USAF
Department of Operational Sciences (ENS)
Air Force Institute of Technology

## MOTIVATION & RESEARCH

US Air Force recruiters routinely assign new recruits to available jobs every month. The goal is to find the best assignments in an efficient manner. Although this problem is modeled as a bipartite assignment matching problem, it is not new to the field of Operations Research. This research presents a new approach to solve assignment matching problems given multiple side constraints. Using two multi-criteria optimization techniques, lexicographic optimization and the elastic constraint method, the assignment matching algorithm efficiently produces an optimal solution in a fraction of the time currently spent. This approach is demonstrated in assigning new USAF recruits to available jobs in any given month for the 369th Recruiting Squadron. The current 369th process manually creates assignments and can take weeks to complete, whereas the assignment matching algorithm takes less than two seconds and, on average, shows an increase in user defined goals of at least 15%.

## METHODOLOGY

•For the 369th Recruiting Squadron:

1st First, determine the maximum number of matches for hard-to-fill jobs

2nd Second, determine the maximum number of matches for each Class

3rd Third, use the determined information to solve the main problem of maximizing the total value of all matches

## INPUTS

Recruits $r \in R$
Jobs $j \in J$
Value $v_{r,j}$
For $r = 1, 2, \ldots, n$
For $j = 1, 2, \ldots, m$

⟷ $= D_{r,j}$ (availability of the recruit to fill the job)

$v_{r,j}$ = Value of the recruit for the job
= Sum of Classes the recruit belongs to plus the top 5 job preferences the recruit lists (if present in the jobs for the given month) plus 1

$w_c$ = Weight for a given Class bounded above and below by the number of Classes and 0, respectively

○ = Hard-to-fill Jobs

## MODEL FORMULATIONS

1st

$$\max F = \sum_{r\in R}\sum_{j\in H} D_{r,j}E_{r,j}$$
subject to
$$\sum_{r\in R} D_{r,j}E_{r,j} \le 1 \quad \forall j\in H$$
$$\sum_{j\in H} D_{r,j}E_{r,j} \le 1 \quad \forall r\in R$$
$$E_{r,j}\in\{0,1\} \quad \forall r\in R,\ j\in H$$

Sub-problem 1 for Hard-to-fill Jobs

2nd

$$\max T_c = \sum_{r\in R}\sum_{j\in J} Rec_{r,c}D_{r,j}E_{r,j} \quad \forall c\in C$$
subject to
$$\sum_{r\in R} D_{r,j}E_{r,j} \le 1 \quad \forall j\in J$$
$$\sum_{j\in J} D_{r,j}E_{r,j} \le 1 \quad \forall r\in R$$
$$E_{r,j}\in\{0,1\} \quad \forall r\in R,\ j\in J$$

Sub-problem 2 for different Classes

3rd

$$\max \sum_{r\in R}\sum_{j\in J} v_{r,j}D_{r,j}E_{r,j} - \sum_{c\in C} w_c s_c$$
subject to
$$\sum_{r\in R} D_{r,j}E_{r,j} \le 1 \quad \forall j\in J$$
$$\sum_{j\in J} D_{r,j}E_{r,j} \le 1 \quad \forall r\in R$$
$$\sum_{r\in R}\sum_{j\in H} D_{r,j}E_{r,j} \ge F \quad \text{(Sub-problem 1)}$$
$$\sum_{r\in R}\sum_{j\in J} Rec_{r,c}D_{r,j}E_{r,j} + s_c \ge T_c \quad \forall c\in C \text{ (Sub-problem 2)}$$
$$E_{r,j}\in\{0,1\} \quad \forall r\in R,\ j\in J$$
$$s_c \ge 0 \quad \forall c\in C$$

Main Problem Formulation

## USER INTERFACE

Step 1 Select Folder Directory
Step 2 Get Recruit/Job Information File
Step 3 Get AFSC Qualifications File
Step 4 Build Necessary Inputs
Step 5 Run Matching Algorithm

Actual / Total
Actual / Max

## CONTRIBUTIONS

• Complexity of the formulation remains the same regardless of the number of objectives (i.e., a polynomial time algorithm)
• Matching algorithm that exceeds (or meets) goals/targets set for each Sub-problem 2 objective
• A new method that significantly decreases the number of man-hours spent to create assignment matches versus the current 369th method
• An optimal set of assignment matches found in an efficient manner
• Matching algorithm that is applicable in many other assignment matching instances

## OUTPUT & RESULTS

| Actual Recruit | Actual Job |
|---|---|
| Applicant 2 | 9T440 |
| Applicant 3 | 9T440 |
| Applicant 5 | 1C131 |
| Applicant 7 | 2A531 |
| Applicant 12 | 9T440 |

•
•
•

Actual 369th problem instance from February 2009

| Statistics | Class 1 (Hisp) | Class 2 (AA) | Class 3 (QT Score) |
|---|---|---|---|
| % of Value | 34.59% | 8.13% | 57.28% |
| (Actual # Assigned)/(Total # Assignments) | 42.99% | 11.21% | 89.72% |
| (Actual # Assigned)/(Max # Matchable) | 80.70% | 66.67% | 97.96% |
| Total # in Class | 111 | 21 | 180 |
| Actual # Assigned | 46 | 12 | 96 |
| Max # in Class Assignable | 57 | 18 | 98 |

| | |
|---|---|
| Total # Assignments | 107 |
| Total # Jobs | 109 |
| % Jobs Assigned | 98.17% |

Results for the Monthly Instances (over total assignments)

| | Dec 2008 | Feb 2009 | Mar 2009 | Apr 2009 | Average |
|---|---|---|---|---|---|
| Class 1 (Hisp.) | 42.30% | 42.99% | 54.05% | 51.04% | 47.60% |
| Class 2 (AF Am.) | 13.00% | 11.21% | 12.61% | 12.50% | 12.83% |
| Class 3 (CAT I-IIIA) | 71.00% | 88.72% | 81.69% | 89.58% | 87.00% |

FY09 Enlisted Accessions Goals/Targets

| | FY09 Goals/Targets |
|---|---|
| Class 1 (Hisp.) | 39.22% |
| Class 2 (AF Am.) | 11.04% |
| Class 3 (CAT I-IIIA) | 75.40% |

Average increase of 21.49%, 16.21%, and 15.38% for Classes 1, 2, and 3, respectively

Sponsor
James D. Hunsicker, Lt Col, USAF
369th Recruiting Squadron
Encino, CA

DEPARTMENT OF OPERATIONAL SCIENCES

## Bibliography

1. Burkard, Rainer E., and Eranda Cela. "Linear Assignment Problems and Extensions." *American Mathematical Society*, 1999: 1-54.

2. Caseau, Yves, and Francois Laburthe. "Solving Various Weighted Matching Problems with Constraints." *Constraints*, 2000: 141-160.

3. Cimen, Zubeyir. *A Multi-Objective Decision Support Model for the Turkish Armed Forces Personnel Assignment System.* WPAFB: AFIT, 2001.

4. Colucci, Simona, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M Donini, Marina Mongiello, and Giacomo Piscitelli. "Semantic-based Approach to Task Assignment of Individual Profiles." *Journal of Universal Computer Science*, 2004: 723-731.

5. DoD. "Population Representation in the Military Services." In *Chapter 2: Active Component Enlisted Applicants and Accessions*, by DoD, 1-27. 2002.

6. Ehrgott, Matthias. *Multicriteria Optimization.* 2nd. New York: Springer, 2005.

7. Ehrgott, Matthias, and David Ryan. "Constructing robust crew schedules with bicriteria optimization." *Journal of Multi-Criteria Decision Analysis* 11, no. 3 (2002): 139-150.

8. Garrett, Deon, Joseph Vannucci, Rodrigo Silva, and Dipankar Dasgupta. "Genetic Algorithms for the Sailor Assignment Problem." *Associations for Computing Machinery*, 2005: 1921-1928.

9. Gass, Saul I. *Linear Programming: Methods and Applications.* Danvers, Massachusetts: boyd & fraser, 1985.

10. Goemans, Michel X. *Lecture notes on bipartite matching.* February 9, 2009. http://math.mit.edu/~goemans/18433S07/matching-notes.pdf (accessed November 8, 2009).

11. Hsieh, Ai-Jia, Chin-Wen Ho, and Kuo-Chin Fan. "An extension of the bipartite weighted matching problem." *Pattern Recognition Letters*, 1995: 347-353.

12. IMADA. *Department of Mathematics and Computer Science (IMADA).* 2009. http://www.imada.sdu.dk/~marco/Teaching/Fall2009/DM204/Slides/TUM-lau.pdf (accessed February 20, 2010).

13. Kleeman, Mark P, and Gary B Lamont. "The Multi-objective Constrained Assignment Problem." *SPIE*, 2007.

14. Lyle, Amaani. *Air Force fiscal 2010 budget reflects rebalanced priorities.* May 29, 2009. http://www.pittsburgh.afrc.af.mil/news/story.asp?id=123151606 (accessed November 3, 2009).

15. Marsman, Steven C. "Recruiting for 2030: Is the US Air Force Getting the Recruits It Needs for the Future?" *Air & Space Power Journal*, 2009.

16. Phillips, Nancy V. "A Weighting Function for Pre-emptive Multicriteria Assignment Problems." *Journal of the Operational Research Society*, 1987: 797-802.

17. Tyson, Ann Scott. *A Historic Success In Military Recruiting: In Midst of Downturn, All Targets Are Met.* October 14, 2009. http://www.washingtonpost.com/wp-dyn/content/article/2009/10/13/AR2009101303539.html (accessed November 3, 2009).

18. Varian, Hall R. "Bootstrap Tutorial." *Mathematica Journal*, 2005: 768-775.

19. Vempala, Santosh. *The Matching Polytope: Bipartite Graphs.* September 23, 2003. http://www-math.mit.edu/~vempala/18.433/L5.pdf (accessed February 22, 2010).

20. Wolsey, Laurence A. *Integer Programming.* Canada: John Wiley & Sons, Inc., 1998.

| REPORT DOCUMENTATION PAGE | | | *Form Approved* *OMB No. 074-0188* |
|---|---|---|---|
| The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.** | | | |
| **1. REPORT DATE** *(DD-MM-YYYY)* 25-03-2010 | **2. REPORT TYPE** Master's Thesis | | **3. DATES COVERED** *(From – To)* Aug 2009 - Mar 2010 |
| **4. TITLE AND SUBTITLE** A MULTI-OBJECTIVE APPROACH TO A BIPARTITE ASSIGNMENT MATCHING PROBLEM USING WEIGHTED VALUES FROM MULTIPLE CONTRAINTS | | | **5a. CONTRACT NUMBER** |
| | | | **5b. GRANT NUMBER** |
| | | | **5c. PROGRAM ELEMENT NUMBER** |
| **6. AUTHOR(S)** Jeong, Greg, S., Captain, USAF | | | **5d. PROJECT NUMBER** |
| | | | **5e. TASK NUMBER** |
| | | | **5f. WORK UNIT NUMBER** |
| **7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)** Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Street, Building 642, WPAFB OH 45433-7765 | | | **8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-OR-MS-ENS-10-05 |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** 369th Recruiting Squadron, Attn: Lt Col James D. Hunsicker, 6337 Balboa Blvd, Ste 5, Encino, CA 91316; COMM: (818) 300-0740; e-mail: James.Hunsicker@Randolph.AF.Mil | | | **10. SPONSOR/MONITOR'S ACRONYM(S)** |
| | | | **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** |
| **12. DISTRIBUTION/AVAILABILITY STATEMENT** APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. | | | |
| **13. SUPPLEMENTARY NOTES** | | | |

**14. ABSTRACT**

US Air Force recruiters routinely assign new recruits to available jobs every month. The goal is to find the best assignments in an efficient manner. Although this problem is modeled as a bipartite assignment matching problem, it is not new to the field of Operations Research. This research presents a new approach to solve assignment matching problems given multiple side constraints. Using two multi-criteria optimization techniques, lexicographic optimization and the elastic constraint method, the assignment matching algorithm efficiently produces an optimal solution in a fraction of the time currently spent. This approach is demonstrated in assigning new USAF recruits to available jobs in any given month for the 369th Recruiting Squadron. The current 369th process manually creates assignments and can take weeks to complete, whereas the assignment matching algorithm takes less than two seconds and, on average, shows an increase in user defined goals of at least 15%.

**15. SUBJECT TERMS**

Multi-objective, Multi-criteria Optimization, Bipartite, Matching, Assignment Matching, Linear Programming, Visual Basic for Applications (VBA), LINGO, Weighted Values, Weighted Matching

| **16. SECURITY CLASSIFICATION OF:** | | | **17. LIMITATION OF ABSTRACT** | **18. NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON** Jeffery D. Weir, Civ, USAF (ENS) |
|---|---|---|---|---|---|
| **a. REPORT** | **b. ABSTRACT** | **c. THIS PAGE** | | | **19b. TELEPHONE NUMBER** *(Include area code)* |
| U | U | U | UU | 94 | (937) 255-3636, ext 4523; e-mail: Jeffery.Weir@afit.edu |

**Standard Form 298 (Rev. 8-98)**
Prescribed by ANSI Std. Z39-18