1-2005

Navy Personnel Planning and the Optimal Partition

Allen G. Holder
Trinity University, aholder@trinity.edu

Navy Personnel Planning and the Optimal Partition

Allen Holder[†]

March 7, 2002

Abstract

One could argue that the Navy's most important resource is its personnel, and as such, workforce planning is a crucial task. We investigate a new model and solution technique that is designed to aid in optimizing the process of assigning sailors to jobs. This procedure attempts to achieve an increased level of sailor satisfaction by providing a list of possible jobs from which a sailor may choose. We show that the optimal partition provided by an interior-point algorithm is particularly useful when designing the job lists. This follows because a strictly complementary solution to the linear programming relaxation observes all possible optimal solutions to the original binary problem. The techniques developed rely on a continuous parametric analysis, and we show that the parameterization provides meaningful information about the structure of the optimal assignments.

Key words: Assignment Problems, Interior Point Algorithms, Matchings, Optimal Partition, Parametric Analysis

---

[†] The Hearin Center of Enterprise Science, School of Business Administration, The University of Mississippi, MS USA. This research was supported by ONR Grant N00014-01-1-0917.

1 Introduction

Assignment problems have long been studied by those working in Operations Research (OR), and these problems are attractive because of their many practical uses and because their structure allows them to be solved efficiently with a simplex based solver. The reason that a simplex algorithm is capable of solving an assignment problem is because the extreme points of the feasible region are binary, a result that is well established.

The fact that some integer programs, like the assignment problem, may be solved without the integer restriction has historically been important because it allows the relaxed linear program (*LP*) to be solved instead of the more difficult integer program. The reason that the simplex algorithm is specifically useful is that it terminates with an extreme point solution, which is integer valued because all extreme points are integer valued. However, most interior-point algorithms terminate with a solution that is characteristically different from the basic optimal solutions provided by a simplex procedure. The difference is that a path-following-interior-point (PFIP) algorithm terminates in the strict interior of the optimal region instead of on the boundary of the optimal region. This means that if there are alternative optimal solutions, a PFIP-algorithm is not guaranteed to terminate with an integer valued solution of the relaxed *LP*. Hence, this 'solution' is not necessarily a solution to the original integer program. In fact, if the original problem is binary, as in the case of the assignment problem, the PFIP-solution is guaranteed not to be binary if there are alternative optimal solutions.

While this may appear to be a limitation of a PFIP-algorithm, we show that there are instances where it is desirable to have both alternative optimal solutions and a PFIP-solution. In fact, as discussed below a PFIP-solution 'observes' all possible solutions, and we use this fact to our advantage to solve an assignment problem for the United States Navy. Moreover, we show that a parametric analysis of the relaxed *LP* provides useful information about the optimal assignments.

2 Notation, Terminology, and Preliminary Results

Many of the results discussed in this section are well established within the interior point community, and we refer interested readers to the three texts of Roos, Terlaky, and Vial [14], Wright [15], and Ye [16]. Also, the terms used in this paper are consistent with the *The Mathematical Programming Glossary* [5]. Consider the standard form linear program and its associated dual,

$$LP: \quad \min\{c^T x : Ax = b, x \geq 0\} \quad \text{and} \quad LD: \quad \max\{b^T y : A^T y + s = c, s \geq 0\},$$

where $A \in \mathbb{R}^{m \times n}$ has full row rank, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$. The primal and dual feasible regions are $\mathcal{P}$ and $\mathcal{D}$, and the corresponding optimal sets are $\mathcal{P}^*$ and $\mathcal{D}^*$. The *strict interiors* of the feasible regions are $\mathcal{P}^o = \{x \in \mathcal{P} : x > 0\}$ and $\mathcal{D}^o = \{(y, s) \in \mathcal{D} : s > 0\}$, and the necessary and sufficient Lagrange conditions for *LP* and *LD* are

$$Ax = b,\ x \geq 0,\ A^T y + s = c,\ s \geq 0, \quad \text{and} \quad x^T s = 0.$$

We make the notational convention that capitalizing a vector variable denotes the diagonal matrix whose diagonal is comprised of the elements of the vector. For example, X is the diagonal matrix with diagonal elements $x_1, x_2, \ldots, x_n$. We also let e be the vector of ones, where length is decided by the context with which it is used. The idea behind a PFIP-algorithm is to replace the complementarity constraint $x^T s = 0$ with $Xs = \mu e$, where μ is a

positive scalar. Notice that μ being positive implies that both x and s are positive. Hence, the primal and dual strict interiors must be non-empty, and we assume throughout that this is the case. The tacit assumption that A has full rank implies that the system,

$$Ax = b,\ x \geq 0,\ A^T y + s = c,\ s \geq 0,\ \text{ and }\ Xs = \mu e,$$

has a unique solution for every positive μ. This solution is denoted by $(x(\mu), y(\mu), s(\mu))$, and the set of all such solutions forms the *central path*. A result first proved by McLinden [11, 12] is that $(x(\mu), y(\mu), s(\mu))$ converges to the *analytic center* of the optimal set as $\mu \downarrow 0$. To understand what this means, we first define the *optimal partition*. This partition is denoted by $(B|N)$ and is defined by

$$B = \{i : x_i > 0 \ \text{ for some } \ x \in \mathcal{P}^*\} \quad \text{and} \quad N = \{1, 2, \ldots, n\} \backslash B.$$

The set B indexes the entire collection of decision variables that are allowed to be positive in an optimal solution, and N indexes the decision variables that are zero in every optimal solution. Allowing a set subscript on a vector (matrix) to be the subvector (submatrix) whose components (columns) are indexed by the set, we have that the optimal partition is important because it characterizes the optimal sets as indicated below,

$$\begin{aligned} \mathcal{P}^* &= \{x \in \mathcal{P} : x_N = 0\} = \{x : A_B x_B = b, x_B \geq 0, x_N = 0\} \quad \text{and} \\ \mathcal{D}^* &= \{(y, s) \in \mathcal{D} : s_B = 0\} = \{(y, s) : A_B^T y = c_B, A_N^T y + s_N = c_N, s_N \geq 0, s_B = 0\}. \end{aligned}$$

The strict interiors of the optimal sets are $(\mathcal{P}^*)^o = \{x \in \mathcal{P}^* : x_B > 0\}$ and $(\mathcal{D}^*)^o = \{(y, s) \in \mathcal{D}^* : s_N > 0\}$. Any solution (x^*, y^*, s^*) such that $x^* \in (\mathcal{P}^*)^o$ and $(y^*, s^*) \in (\mathcal{D}^*)^o$ is called *strictly complementary*, and it has been known that every well-posed linear program has such a solution since 1956 [4]. The analytic centers of the primal and dual optimal sets, denoted by x^c and (y^c, s^c), are the unique solutions to

$$\max \left\{ \sum_{i \in B} \ln(x_i) : x \in (\mathcal{P}^*)^o \right\} \quad \text{and} \quad \max \left\{ \sum_{i \in N} \ln(s_i) : (y, s) \in (\mathcal{D}^*)^o \right\}.$$

McLinden [11, 12] was able to show that $x(\mu) \to x^c$ and $(y(\mu), s(\mu)) \to (y^c, s^c)$, as $\mu \downarrow 0$. This result is significant because x^c and (y^c, s^c) are strictly complementary. Since PFIP-algorithms follow the central path towards optimality, this means that the solution to *LP* and *LD* produced by a PFIP-algorithm is in the strict interior of the optimal set. Subsequently, if (x^*, y^*, s^*) is the terminal iterate of a PFIP-algorithm, we have that $x_B^* > 0$, $x_N^* = 0$, $s_N^* > 0$, and $s_B^* = 0$. So, we can glean the optimal partition from the PFIP solution by distinguishing which components are zero and which are positive.

What is not found in the literature is that there is a special interpretation of the optimal partition for binary problems. From this point on we assume that A and b are such that the extreme points of $\mathcal{P}$ are binary valued. Consider the binary linear program and its relaxation,

$$\begin{aligned} \textit{BLP}: \quad & \min\{c^T x : Ax = b, x_i \in \{0, 1\}, i = 1, 2, \ldots, n\} \\ \text{and} \quad \textit{LPR}: \quad & \min\{c^T x : Ax = b, 0 \leq x \leq e\}. \end{aligned}$$

We let

$$d = e - x, \quad z = \begin{pmatrix} x \\ d \end{pmatrix}, \quad h = \begin{pmatrix} b \\ e \end{pmatrix}, \quad u = \begin{pmatrix} c \\ 0 \end{pmatrix}, \quad \text{and} \quad H = \begin{bmatrix} A & 0 \\ I & I \end{bmatrix},$$

so that the standard form of *LPR* is

$$LPR': \quad \min\{u^T z : Hz = h, z \geq 0\}.$$

Let $z^* = ((x^*)^T|(d^*)^T)^T$ be a strictly complementary solution to *LPR'* and $(B|N)$ be the optimal partition of *LPR'*. Because z^* contains both x^* and d^* as subvectors, the optimal partition indexes components of both x^* and d^*. We let B' and B'' partition B such that $0 < z^*_B = ((x^*_{B'})^T|(d^*_{B''})^T)^T$. Similarly, N' and N'' partition N such that $0 = z^*_N = ((x^*_{N'})^T|(d^*_{N''})^T)^T$. We further partition B' into U' and M' and B'' into U'' and M'' by defining

$$U' = \{i \in B' : x^*_i = 1\}, \quad M' = B' \backslash U', \quad U'' = \{i \in B'' : d^*_i = 1\}, \quad \text{and} \quad M'' = B'' \backslash U''.$$

From these definitions and the fact that $e = x + d$, we have that

$$e = \begin{pmatrix} x_{U'} \\ x_{M'} \\ x_{N'} \end{pmatrix} + \begin{pmatrix} d_{N''} \\ d_{M''} \\ d_{U''} \end{pmatrix}. \tag{1}$$

Theorem 1 shows that the special nature of *BLP* implies that the components indexed by U' have a value of one in every optimal solution. Moreover, the result states that the elements in M' can be 0 or 1 in an optimal solution.

Theorem 1 *If x^* is an optimal solution to BLP, we have that $x^*_{N'} = 0$ and that $x^*_{U'} = 1$. Moreover, for each $i \in M'$, we have that there exists optimal solutions $\hat{x}^*$ and $\hat{\hat{x}}^*$ such that $\hat{x}^*_i = 0$ and $\hat{\hat{x}}^*_i = 1$.*

Proof: Let x^* be an optimal solution to *BLP*. From the assumption that the extreme points of the feasible region are binary valued, we have that the optimal set of *BLP* is contained in the optimal set of *LPR*. So, setting $d^* = e - x^*$, we have that $z^* = ((x^*)^T|(d^*)^T)^T$ is an optimal solution to *LPR'*. Since $N = N' \cup N''$, we have from the definition of N that $x^*_{N'} = 0$ and that $d^*_{N''} = 0$. We subsequently have from (1) that $x^*_{U'} = 1$.

Now, let $z^* = ((x^*)^T|(d^*)^T)^T$ be a strictly complementary solution to *LPR'*. The assumption that the strict interiors of the feasible regions are non-empty is equivalent to the optimal sets being bounded [14]. Hence, we have that z^* is the convex combination of the extreme points of the optimal set. This means that $x^*_{M'}$ is the convex combination of binary vectors. Since $0 < x^*_{M'} < e$, we conclude that for each $i \in M'$ that there exists optimal solutions $\hat{x}^*$ and $\hat{\hat{x}}^*$ such that $\hat{x}^*_i = 0$ and $\hat{\hat{x}}^*_i = 1$. ■

In a sense, Theorem 1 states that all possible solutions to the binary problem are observed from a single strictly complementary solution to the relaxed problem. For example, suppose we solve *LPR* with a PFIP-algorithm and have the strictly complementary solution x^*. From Theorem 1 we have that the i^{th} element of every optimal solution to the binary problem is either 0 or 1 if x^*_i is either 0 or 1, respectively. Moreover, if x^*_i is between 0 and 1, we know that there are optimal solutions to the binary problem where the i^{th} component is 0 in one solution and 1 in another. So, while a strictly complementary solution to the *LP* relaxation may not be a solution to the binary problem, we do in a sense see every optimal solution. We use this to our advantage in the following sections.

One of the analysis results developed in Section 4 depends on the existence of a matching in a subgraph of a bipartite graph, and we use Hall's famous Matching Theorem [6]. This result is particularly useful because it characterizes the bipartite graphs that permit a matching. If $G = (V, E)$ is a graph, the *neighborhood* of $v \in V$ is $N(v) = \{w \in V : (v, w) \in E\}$, and if $W \subseteq V$, the neighborhood of W is $N(W) = \cup_{w \in W} N(w)$.

Theorem 2 (Hall [6]) *Let (V_1, V_2, E) be a bipartite graph. The elements in V_1 can be matched to the elements in V_2 if, and only if, for any $W \subseteq V_1$ we have that $|W| \leq |N(W)|$.*

The result that we use is a corollary of Hall's Theorem. Let $G = (V_1, V_2, E)$ be a bipartite graph such that the elements in V_1 may be matched to the elements in V_2. We define $K(G)$ so that $K(G) = \min\{|N(W)| - |W| : W \subseteq V_1\}$.

Corollary 1 *Let $G = (V_1, V_2, E)$ be a bipartite graph, and assume that the elements in V_1 can be matched to the elements in V_2. Let $W \subseteq V_2$ be such that $|W| \leq K(G)$. Then, for $F = \{(v, w) : w \in W\}$ we have that the bipartite graph $G' = (V_1, V_2 \backslash W, E \backslash F)$ has the property that the elements in V_1 can be matched to the elements in $V_2 \backslash W$.*

Proof: The fact that $|W| \leq K(G)$ guarantees that the subgraph G' has the property that $|Y| \leq |N(Y)|$ for every $Y \subseteq V_1$ —i.e. we have decreased the size of any neighborhood of V_1 by at most $K(G)$. So, the result follows immediately from Hall's Matching Theorem. ■

3 An Assignment Problem

In this section we develop an assignment problem that is used for workforce planning. This model was inspired by the needs of the United States Navy, and a brief description of their situation is warranted. There are roughly 300,000 sailors in the Navy, and about 120,000 of these sailors change, or *rotate*, jobs every year. The people in charge of handling these job re-assignments are called detailers. The assignment process is complex not just because of the volume of re-assignments, but also because the detailer is trying to satisfy the often contradictory desires of the sailor and the Navy. Because these problems are complex, the Navy has experimented with several automated techniques designed to aid the detailers, see [1, 2, 8, 9, 10].

Every two weeks a detailer receives an updated list of jobs that need to be filled, and their job is to place the sailors going through their job rotation into these jobs as best they can. While a detailer's primary concern is to make sure that the Navy's staffing requirements are satisfied, they must also attempt to achieve a high level of sailor satisfaction. This is a difficult task because of the manner in which the re-assignment process is undertaken. When a sailor contacts a detailer, the detailer locates the jobs that the sailor is capable of filling (this is discussed in detail below) and most often selects a job for the sailor. The detailer's job selection may, or may not, depend on any preferences provided by the sailor, and it is often the case that the sailor is disgruntled with their new assignment. However, the detailer is making a decision based on the sailor's skills and the current desires of the Navy. Consider the following scenario. Suppose that there are two jobs that are well suited for a sailor, say one job was geographically preferred by the sailor but required some extra training and another job was in a less desirable location but matched the sailor's skill set. Overall, the sailor is indifferent to which job is assigned, and the detailer decides to assign the sailor to the job having the desirable location. However, the next sailor that calls has the same geographic preference and has the skills to start the job without training, but because this job was just filled, the new sailor can not be assigned to the job. This means that the new sailor may be assigned to a job that 1) requires extra training (and expense to the Navy) and 2) is not in a geographically preferred location.

The problem here is that the detailer did not have any knowledge about which sailor would call next. Each sailor has a Projected Rotation Date (PRD), and a sailor must be assigned to

a job from nine to six months prior their PRD. However, the list of available jobs, called the requisition list, is updated every two weeks. So, it is possible for a sailor to observe several requisition lists. When a sailor contacts a detailer to discuss career possibilities, we need to locate a job for this sailor, but at the same time we need to 'reserve' jobs for the sailors that may still call (we refer to this as *hedging*). Moreover, we want to do this in an optimal manner. As an added complication, to attain an increased level of sailor satisfaction the Navy wants each sailor to be presented with a list of possible jobs. Most sailors will require some time to think about their options, and we assume that the jobs offered to a sailor can not be offered to subsequent sailors for a brief period of time (something like 3 or 4 days). The assignment model and solution procedures developed below are directed at generating job lists in this scenario.

We assume that we are planning for a time period of length T (two weeks in the Navy application). We let J be the set of jobs on the requisition list and S be the set of sailors whose PRD is within six to nine months past the last day of the time period. Typically, only a small portion of the sailors will request a job list during the planning period, but again, we do not know which of the sailors will contact their detailer. We index the sailors by i and the jobs by j, and define $\mathcal{A} = \{(i,j) : \text{ it possible for sailor } i \text{ to do job } j\}$. Also, for each $i \in S$ we let $\mathcal{A}_i = \{j : (i,j) \in \mathcal{A}\}$, and for each $j \in J$ we let $\mathcal{A}_j = \{i : (i,j) \in \mathcal{A}\}$. Whether or not job j can appear on sailor i's job list depends on four criteria. First, sailor i must have the correct rank and pay grade required for job j. Second, each sailor must begin their new job within the seven month time period of three months before the PRD to four months after the PRD. So, job j can appear on sailor i's job list only if the starting time for job j is within this time period. Third, sailors are required to rotate from sea duty to shore duty and from shore duty to sea duty. So, job j must have the correct sea/shore designation for it to be a candidate for sailor i's job list. Fourth, each job has a list of required Navy Enlisted Classifications (NECs), and sailor i can fill job j only if they either have the required skills or can attain them en route to the new job. Some jobs may be filled by any sailor, such as a recruiter, and there are always enough of these jobs on the requisition list so that for any collection of sailors $S' \subseteq S$, we have that $|S'| \leq |\cup_{i \in S'} A_i|$. So, we have from Hall's Matching Theorem that the sailors may always be matched to jobs.

The most interesting and scrutinized part of the model is the objective function. This is because a detailer's job is complex, and the objective coefficients attempt to capture a detailer's expertise. As previously stated, the detailer's first responsibility is to guarantee that the Navy's staffing requirements are satisfied. There are two manners in which the Navy manifests its workforce requirements. First, they control the jobs that are released for assignment, and second, each job that is released is given a priority value between 0 and 100. A detailer only sees the released jobs, and hence, is only concerned with a job's priority. Moreover, high priority jobs do not have to be filled as soon as possible, but rather these jobs should be filled over lower priority jobs when all other factors are about the same.

In addition to the job priority, a detailer is to consider the cost of the re-assignment and the desires of the sailor. The moving cost of a re-assignment is indirectly considered through the attempt to keep a sailor based in the same location, and in general, sailors prefer this for their families. For example, a sailor could complete a shore job at Norfolk, VA and then be assigned to sea duty on a ship based out of this port. Educational costs are kept in check by matching the sailor's NECs to the job NECs as best as possible. Also, even if it were possible for a sailor to acquire two or more NECs as they rotate to their new job, this would be discouraged (in fact, most sailors only receive one or two NECs their entire career). So, in general it is

desirable for a sailor to obtain at most one new NEC per rotation. The only information that a detailer has about a sailor's desires prior to discussing a re-assignment is a list of geographical preferences (this is called a 'dream sheet' and lists three preferred locations for sea, shore, and overseas duty). The preferences listed are questionable because sailors have little or no information when they fill out their dream sheet and because sailors believe that detailers do not consider these preferences when assigning jobs. However, even with these discrepancies we include the sailor preferences in our model. This is consistent with the Navy's current desire to increase sailor satisfaction, and until this information is considered, the (mis)conception that a sailor's preferences are not important will remain.

We use a scoring technique to decide the objective coefficients. The score for each possible assignment is the sum of four subscores that measure training, location, Navy priority, and geographical preference. Specifically, for all $(i,j) \in \mathcal{A}$ and $j \in J$ we define

$$\begin{aligned}
T_{(i,j)} &= \begin{cases} 1, & \text{sailor } i \text{ lacks 2 or more NECs required by job } j \\ 0, & \text{otherwise,} \end{cases} \\
L_{(i,j)} &= \begin{cases} 1, & \text{job } j\text{'s location is different from sailor } i\text{'s current location} \\ 0, & \text{otherwise,} \end{cases} \\
P_j &= \begin{cases} 1, & \text{job } j\text{'s priority is lower than the average priority} \\ 0, & \text{otherwise,} \end{cases} \\
G_{(i,j)} &= \begin{cases} 1, & \text{job } j\text{'s location is not preferred by sailor } i \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}$$

Each of these functions penalizes an undesirable situation by contributing a one to the possible assignment, and the total score, or penalty, for each $(i,j) \in \mathcal{A}$ is

$$c_{(i,j)} = T_{(i,j)} + L_{(i,j)} + P_j + G_{(i,j)}.$$

The interpretation of these scores is easy. The possible assignment (i,j) scores a zero if sailor i is at job j's location, the sailor wants to stay at that location, the sailor is (nearly) trained for the job, and the Navy has placed a high priority on the job. The assignment scores a four if sailor i is not at job j's location, the sailor does not want to move to that location, the sailor is not trained for the position, and the Navy has given the job a low priority.

For each $(i,j) \in \mathcal{A}$, we set $x_{(i,j)}$ to be a binary decision variable that indicates if sailor i is 'assigned' to job j (as discussed momentarily, we use the word assigned in a liberal manner here). The assignment problem that we consider is

$$\begin{aligned}
\mathcal{AS}: \quad \min \quad & \sum_{(i,j)\in\mathcal{A}} c_{(i,j)} x_{(i,j)} \\
\text{s.t.} \quad & \sum_{j\in\mathcal{A}_i} x_{(i,j)} \geq 1, \text{ for } i \in S, \\
& \sum_{i\in\mathcal{A}_j} x_{(i,j)} \leq 1, \text{ for } j \in J, \\
& x_{(i,j)} \in \{0,1\}, \text{ for } (i,j) \in \mathcal{A}.
\end{aligned}$$

The constraints guarantee that each job is assigned to at most one sailors and that each sailor is 'assigned' at least one job. This is not a matching problem because a sailor may be 'assigned' several jobs. However, recall that we are not actually making assignments but are instead generating job lists. We also point out that the corresponding coefficient matrix is

totally unimodular, which subsequently implies that the extreme points of the feasible region are binary valued.

There are two reasons why we use the simple zero/one scoring imposed by $T_{(i,j)}$, $L_{(i,j)}$, P_j, and $G_{(i,j)}$. First, we discovered after numerous discussions with the detailers that each assignment is unique, and the subtle mediations that occur between a sailor and a detailer would be impossible to capture in a single model that is to be used in every situation. So, while a more refined model might apply to many of the typical situations, the crude zero/one scoring that we use captures the basic concepts sought after in all situations. Second, we want a model with alternative optimal solutions. This follows because we want to generate a list of jobs for each sailor. The fewer possible values that $c_{(i,j)}$ can attain, the more likely it is that there are multiple optimal solutions. For example, if all the objective coefficients were the same, every feasible solution would be optimal. In our model we have that each $c_{(i,j)}$ is one of $0, 1, 2, 3$, or 4. Putting these two characteristics together, we interpret *AS* as eliminating bad assignments instead of finding a single optimal assignment.

Before continuing with the development of our solution techniques, we define two important bipartite graphs. The first graph is $G = (S, J, \mathcal{A})$, which represents the feasible region of *AS*. As already mentioned, the requisition list is designed so that the sailors can always be matched to the jobs, which guarantees that *AS* is feasible. Let x^* be a strictly complementary solution to the *LP* relaxation of *AS*, and let $\mathcal{A}^*$ be $\{(i,j) \in \mathcal{A} : x^*_{(i,j)} > 0\}$ —i.e. $\mathcal{A}^*$ is $B' = U' \cup M'$ in (1). From Theorem 1 we have that $\mathcal{A}^*$ is the collection of optimal assignments, meaning that (i,j) is in $\mathcal{A}^*$ if, and only if, there is an optimal solution to *AS* where sailor i is assigned to job j. The second graph is $G^* = (S, J, \mathcal{A}^*)$, and this graph represents the collection of optimal assignments. In the next section we develop two techniques that use G^* to generate job lists as sailors contact their detailers.

4 Solution Techniques

In this section we show how to use parametric analysis to generate a job list. The first step is to solve the *LP* relaxation of *AS* with a PFIP-algorithm and generate $\mathcal{A}^*$. From Theorem 1 we have that any assignment with a positive value can be optimal. A naive way to generate a job list for sailor i is to find every job j such that $(i,j) \in \mathcal{A}^*$. However, this naive approach provides little control over the list length. For example, there might be a single job j such that $(i,j) \in \mathcal{A}^*$, and hence, this naive approach generates a list of size one. As a further example, suppose that the first three sailors and jobs in G^* are illustrated in Figure 1. Observe that $(i,1) \notin \mathcal{A}^*$ for $i = 2, 3$, which means that it is never optimal for sailor 2 or 3 to be assigned to job 1. Also, jobs 1 and 2 are optimal for sailor 1 only, and hence, these jobs should appear on sailor 1's job list. However, each of the three sailors could be assigned to job 3 optimally, and the question becomes how do we decide if job 3 should appear on a sailor's job list.

Suppose that sailor $\hat{i}$ has contacted a detailer and is asking about possible re-assignments.

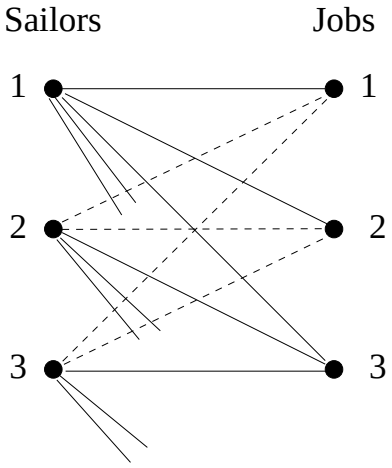


Figure 1: The dashed lines indicate that (i,j) is not in $\mathcal{A}^*$ and the solid lines indicate that (i,j) is in $\mathcal{A}^*$.

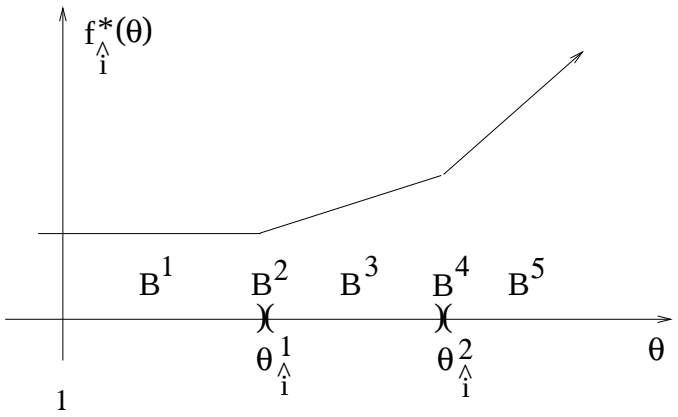


Figure 2: How the linearity intervals of $f^*_{\hat{i}}$ correspond to the the optimal partitions.

We use the following parametric model to generate the job list.

$$
\begin{aligned}
ASP_\theta: \quad \min \quad & \sum_{(i,j)\in\mathcal{A}} c_{(i,j)} x_{(i,j)} \\
\text{s.t.} \quad & \sum_{j\in\mathcal{A}_i} x_{(i,j)} \geq 1, \text{ for } i \in S\backslash\{\hat{i}\} \\
& \sum_{j\in\mathcal{A}_{\hat{i}}} x_{(\hat{i},j)} \geq \theta, \\
& \sum_{i\in\mathcal{A}_j} x_{(i,j)} \leq 1, \text{ for } j \in J \\
& x_{(i,j)} \in \{0,1\}, \text{ for } (i,j) \in \mathcal{A}.
\end{aligned}
$$

While the list length for sailor $\hat{i}$ is controlled by the value of θ, this parameter is not to be interpreted as the actual list length. The reason for this is that alternative optimal solutions allow the list length to be greater than θ. Consider the partial solution depicted in Figure 1 for $\theta = 1$. Although θ is 1, it is clear that jobs 1 and 2 should appear on sailor 1's job list. An appropriate interpretation of θ is that it guarantees a minimum list length. The difference between the list length and θ is important because the approaches below find the smallest θ that provides a sufficient list length. Keeping θ as small as possible makes intuitive sense because as θ increases, sailor $\hat{i}$ is receiving more and more job choices. So, large values of θ restrict the possibilities for subsequent sailors. Since we want future sailors to receive adequate job lists, keeping θ small is important.

Deciding on a target list length is not an easy task. First, the length of the list is being used to achieve sailor satisfaction, and it makes sense that longer list lengths should be used as a reward for well performing sailors. Also, there is an incentive to provide sailors near the end of their contracts more choices because these sailors may decide to leave the Navy if a suitable job is not found. Second, the number of alternative optimal solutions partially decides the list length. To see how this works, suppose that the list length is chosen to be 3 and that for $\theta = 1$ there are 5 jobs exclusively optimal for sailor $\hat{i}$ (meaning that sailor $\hat{i}$ could choose any of the 5

jobs and not effect another sailor's list). In this situation it makes sense to offer all 5 jobs and not just 3. The parametric techniques developed below overcome these problems, but they do require a 'guidance' list length. We let w_i be the relative performance score of sailor i (sailor i's performance score divided by maximum possible performance score) and l_i be the percent of sailor i's contract that is fulfilled. We let M be a size decided by the detailer that generally describes the largest list length necessary, and we then use $\hat{M}_i = w_i l_i M$ as a 'guide' for the list length of sailor i (how this value is used is described below). So, sailors that have good performance scores and that are near the end of their contracts receive more job choices.

After adding slack and surplus variables to ASP_θ, we let A, b, and c be such that the LP relaxation of ASP_θ is

$$ASPR_\theta: \quad \min\{c^T z : Az = b + \theta e_{\hat{i}}, z \geq 0\},$$

where $e_{\hat{i}} = (0, \ldots, 0, 1, 0, \ldots, 0)^T$ and the 1 is in the position corresponding to the parametric constraint in ASP_θ. We point out that b has a zero in the position corresponding to the parametric constraint so that $ASPR_1$ is the original AS problem. We assume that d is the slack and surplus vector so that z is $(x^T|d^T)^T$. We note that the constraints in $ASPR_\theta$ include the upper bound $x \leq e$, and that the matrix A is totally unimodular. So, for integer values of θ we have that the extreme points of the feasible region are binary valued.

Our first approach was to solve $ASPR_{\hat{M}_{\hat{i}}}$ with a PFIP-algorithm, glean the optimal partition of $ASPR_{\hat{M}_{\hat{i}}}$, and use this optimal partition to form a job list. However, the optimal partition of $ASPR_{\hat{M}_{\hat{i}}}$ often provided a job list that was longer than necessary. This is an immediate consequence of Theorem 1 since a strictly complementary solution to the LP relaxation observes all optimal solutions. As previously discussed, the problem here is the difference between the list length and the value of θ. To overcome this problem, we have adopted a parametric analysis approach, and we begin with a brief review of how the optimal partition is used to conduct parametric analysis. This analysis was independently developed by Monteiro and Mehrotra [13] and Roos, Terlaky, and Vial [14] (it was also used in [7] to calculate an optimal partition for multiple objective programs). We let $f^*_{\hat{i}}(\theta)$ be the optimal objective value of $ASPR_\theta$, and it is well known that $f^*_{\hat{i}}(\theta)$ is piecewise linear, continuous, and convex [3] (the $\hat{i}$ subscript indicates which sailor's job list is being constructed). What the authors of [13] and [14] noticed was that the linearity intervals of $f^*_{\hat{i}}(\theta)$ correspond with a unique optimal partition. For example, suppose that as θ increases from one, $f^*_{\hat{i}}(\theta)$ is the function illustrated in Figure 2. The first linearity interval terminates at $\theta^1_{\hat{i}}$, the second linearity interval is from $\theta^1_{\hat{i}}$ to $\theta^2_{\hat{i}}$, and the third linearity interval starts at $\theta^3_{\hat{i}}$. Because $\theta^1_{\hat{i}}$, $\theta^2_{\hat{i}}$, and $\theta^3_{\hat{i}}$ define the linearity intervals of $f^*_{\hat{i}}$, we call them *break points*. We consecutively label the break points of $f^*_{\hat{i}}$ that are at least 1 by $\theta^1_{\hat{i}}, \theta^2_{\hat{i}}, \ldots$. We point out that it is possible for $\theta^1_{\hat{i}}$ to be 1, which means the optimal partition immediately changes as θ increases from 1. In the example, the optimal partition for all θ in the interval $[1, \theta^1)$ is $(B^1|N^1)$ (indicated by B^1 in the figure), but the optimal partition for $\theta^1_{\hat{i}}$ itself is $(B^2|N^2)$. Similarly, the optimal partition for all θ in the open interval $(\theta^1_{\hat{i}}, \theta^2_{\hat{i}})$ is $(B^3|N^3)$, and for $\theta^2_{\hat{i}}$ the optimal partition is $(B^4|N^4)$. This process continues until no break points are found. We note that the optimal partition approach contrasts the traditional theory that relied on basic optimal solutions. In fact, one can not guarantee that there is a basis that remains optimal for an entire linearity interval.

There are two alternating steps to calculate $f^*_{\hat{i}}$. The first step is to find out how far θ can increase before a change in the optimal partition is forced. The answer to this question is found by solving a linear program, and the optimal partition to this linear program induces the optimal partition located at the next break point. The second step is to find the rate at

which $f^*_{\hat{i}}$ changes over the next linearity interval. Again, this rate is calculated by solving a linear program, and the optimal partition for this problem induces the optimal partition for the next linearity interval. The process continues until no break points are found. The details of the process are described in Algorithm 1 for the example in Figure 2.

Algorithm 1

Step 1: Find out how far $(B^1|N^1)$ can remain the optimal partition by solving

$$\max\{\theta : A_{B^1} z_{B^1} - \theta e_{\hat{i}} = b, z_{B^1} \geq 0, \theta \geq 0\}$$

Let $(z^*_{B^1}, \theta^*)$ be a strictly complementary solution. The optimal value is $\theta^1_{\hat{i}}$ —i.e. θ^* is $\theta^1_{\hat{i}}$. The next optimal partition is induced by $z^*_{B^1}$ because

$$B^2 = \{i \in B^1 : z^*_i > 0\} \quad \textit{and} \quad N^2 = N^1 \cup \{i \in B^1 : z^*_i = 0\}.$$

Step 2: Find the rate of change for $f^*_{\hat{i}}(\theta)$ over the next linearity interval by solving

$$\max\{y_{\hat{i}} : A_{B^2} y = c_{B^2}, A_{N^2} y + s_{N^2} = c_{N^2}, s_{N^2} \geq 0\}.$$

The solution is the slope of $f^*_{\hat{i}}$ over the interval from θ^1 to θ^2. Let $(y^*, s^*_{N^2})$ be a strictly complementary solution. Similar to the previous case, this solution induces the next optimal partition,

$$B^3 = B^2 \cup \{i \in N^2 : s^*_i = 0\} \quad \textit{and} \quad N^3 = \{i \in N^2 : s^*_i > 0\}.$$

Step 3: Find out how far $(B^3|N^3)$ can remain optimal by solving

$$\max\{\theta : A_{B^3} z_{B^3} - (\theta^1_{\hat{i}} + \theta) e_{\hat{i}} = b, z_{B^3} \geq 0, \theta \geq 0\}.$$

Let $(z^*_{B^3}, \theta^*)$ be a strictly complementary solution. The optimal value is $\theta^2_{\hat{i}}$, and the next optimal partition is induced by $z^*_{B^3}$ because

$$B^4 = \{i \in B^3 : z^*_i > 0\} \quad \textit{and} \quad N^4 = N^3 \cup \{i \in B^3 : z^*_i = 0\}.$$

Step 4: Find the rate of change for $f^*_{\hat{i}}(\theta)$ over the next linearity interval by solving

$$\max\{y_{\hat{i}} : A_{B^4} y = c_{B^4}, A_{N^4} y + s_{N^4} = c_{N^4}, s_{N^4} \geq 0\}.$$

The solution is the slope of $f^*_{\hat{i}}$ for θ greater than θ^2. Let $(y^*, s^*_{N^4})$ be a strictly complementary solution. Then, $y^*, s^*_{N^4}$ induces the next optimal partition as follows,

$$B^5 = B^4 \cup \{i \in N^4 : s^*_i = 0\} \quad \textit{and} \quad N^5 = \{i \in N^4 : s^*_i > 0\}.$$

Step 5: Continue in a like fashion.

We use Algorithm 1 to find the optimal partitions of $ASPR_\theta$, for integer values of θ up to $\hat{M}_{\hat{i}}$. Let $(B(\theta)|N(\theta))$ be the optimal partition of $ASPR_\theta$, so $(B(1)|N(1))$ is the optimal partition for the original AS problem. Recall that $B(\theta)$ indexes components in $z = (x^T|d^T)^T$, and we let $\mathcal{A}^*(\theta)$ be the collection of indices in $B(\theta)$ that correspond to x. We mention that it is possible for $\hat{\theta} \neq \bar{\theta}$ and $B(\hat{\theta}) \neq B(\bar{\theta})$, but have that $\mathcal{A}(\hat{\theta}) = \mathcal{A}(\bar{\theta})$ —i.e. the slack and surplus

variables can force the change in the optimal partition. While θ is continuous, Theorem 3 shows that the break points of $f^*_{\hat{i}}$ are integer valued.

Theorem 3 *We have that $\theta^k_{\hat{i}}$ is integer valued for $k = 1, 2, \ldots$.*

Proof: For some partition $(\bar{B}|\bar{N})$ and right-hand side $\bar{b}$, the linear program that calculates the range over which a partition remains optimal is

$$\max\{\theta : A_{\bar{B}} z_{\bar{B}} - e_{\hat{i}}\theta = \bar{b},\ z_{\bar{B}} \geq 0, \theta \geq 0\},$$

The coefficient matrix, $[A_{\bar{B}}|e_{\hat{i}}]$, is totally unimodular since A itself is totally unimodular and $e_{\hat{i}}$ is a column of the identity matrix. The first range calculation has $\bar{b} = b$ as its right-hand side, and hence, the right-hand side is binary valued. We conclude that $\theta^1_{\hat{i}}$ is integer valued. The second break point is calculated with the right-hand side $\bar{b} = b + \theta^1_{\hat{i}} e_{\hat{i}}$ (the partition is also different), which is integer valued because b and $\theta^1_{\hat{i}}$ is integer valued. Hence, $\theta^2_{\hat{i}}$ is also integer valued. In general, the k^{th} break point is calculated with the right-hand side $\bar{b} = b + (\sum_{i=1}^{k-1} \theta^i_{\hat{i}}) e_{\hat{i}}$, and because this is integer valued, we have by induction that $\theta^k_{\hat{i}}$ is integer valued. ■

Our goal is to calculate $(B(\theta)|N(\theta))$ for integer values of θ between 1 and $\hat{M}_{\hat{i}}$, and one way to calculate these partitions is to solve $ASPR_1, ASPR_2, \ldots, ASPR_{\hat{M}_{\hat{i}}}$. We instead use the parametric technique in Algorithm 1 for two reasons. First, there is the possibility that the number of linear programs to solve is reduced. For example, suppose that $\theta^1_{\hat{i}}$ is 6. Then we know that $(B(1)|N(1)) = (B(2)|N(2)) = \ldots = (B(5)|N(5))$. Moreover, we have calculated $(B(6)|N(6))$, which is different from the previous partitions because it is located at a break point. Since $\theta^1_{\hat{i}}$ is calculated by solving a single linear program after $ASPR_1$ is solved, we have reduced the number of linear programs from 6 to 2. Second, the parametric technique adds to our analysis capability. Let $\hat{\theta}$ be the first break point that is at least $\hat{M}_{\hat{i}}$. One of the reasons that $\hat{M}_{\hat{i}}$ is used as a guidance value is that there may be several integers in $[\hat{M}_{\hat{i}}, \hat{\theta}]$. Such a situation indicates an indifference to a list length of $\hat{M}_{\hat{i}}$ and list length of $\hat{\theta}$, and hence, sailor $\hat{i}$ may be offered more job choices than originally sought.

The next two results show that $\theta^1_{\hat{i}}$ provides meaningful information about ASP_θ. We define the following sets for any $\hat{i}$.

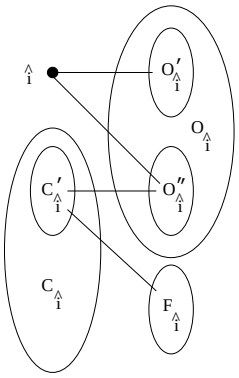
$$\begin{aligned}
O_{\hat{i}} &= \{j : (\hat{i}, j) \in \mathcal{A}^*\},\\
O'_{\hat{i}} &= \{j \in O_{\hat{i}} : (i, j) \in \mathcal{A}^* \Rightarrow i = \hat{i}\},\\
C_{\hat{i}} &= \{i \neq \hat{i} : (i, j) \in \mathcal{A}^* \text{ for some } j \in O_{\hat{i}}\},\\
C'_{\hat{i}} &= \{i \in C_{\hat{i}} : O'_i \neq \emptyset\},\\
F_{\hat{i}} &= \{j : j \in O'_i \text{ for some } i \in C'_{\hat{i}}\},\\
O''_{\hat{i}} &= \{j \in O_{\hat{i}} : (i, j) \in \mathcal{A}^* \Rightarrow i \in \{\hat{i}\} \cup C'_{\hat{i}}\}.
\end{aligned}$$

The set $O_{\hat{i}}$ contains the jobs that may be assigned to sailor $\hat{i}$ optimally, and the set $O'_{\hat{i}}$ contains the jobs that are exclusively optimal for sailor $\hat{i}$. We say that the jobs in $O_{\hat{i}} \backslash O'_{\hat{i}}$ are *contended* by other sailors. A sailor $i \neq \hat{i}$ is in $C_{\hat{i}}$ if there is a job which may be assigned optimally to either sailor i or $\hat{i}$. So, the sailors in $C_{\hat{i}}$ are *competing* with sailor $\hat{i}$ for jobs. For sailor i to be in $C'_{\hat{i}}$ we need the further restriction that sailor i has a uniquely optimal job. This is a favorable

situation since this means that a uniquely optimal job can be 'saved' for this sailor. The set $O''_{\hat{i}}$ contains the jobs that are optimal only for sailor $\hat{i}$ or any of the sailors in $C'_{\hat{i}}$. We say that the jobs in $O'_{\hat{i}}$ and $O''_{\hat{i}}$ are *free choices* for sailor $\hat{i}$ because these jobs may be offered to sailor $\hat{i}$ while still guaranteeing that the other sailors have an optimal assignment. Theorem 4 shows that the number of jobs in $O_{\hat{i}}$, $O'_{\hat{i}}$, and $O''_{\hat{i}}$ bound $\theta^1_{\hat{i}}$.

Theorem 4 *We have that* $|O_{\hat{i}}| \geq \theta^1_{\hat{i}} \geq |O'_{\hat{i}}| + |O''_{\hat{i}}|$.

Proof: Let $z^* = ((x^*)^T|(d^*)^T)^T$ be a strictly complementary primal solution to $ASPR_1$, and y^* be a strictly complementary dual solution. The first inequality is obvious, since θ can not increase past $|O_{\hat{i}}|$ without another job becoming optimal. The optimal partition $(B(1)|N(1))$ remains optimal so long as the following system is consistent



$$A_{B(1)}z_{B(1)} = b + \theta e_{\hat{i}}, \quad z_{B(1)} > 0, \quad A^T_{B(1)}y = c_{B(1)}, \quad \text{and} \quad A^T_{N(1)}y < c_{N(1)}.$$

Since $A^T_{B(1)}y^* = c_{B(1)}$ and $A^T_{N(1)}y^* < c_{N(1)}$, we only need to guarantee the existence of $z_{B(1)}(\theta)$ such that $A_{B(1)}z_{B(1)}(\theta) = b$ and $z_{B(1)}(\theta) > 0$. This means that we must show that the value of $x^*_{\mathcal{A}^*}$ may be adjusted so that no new bounds of $ASPR_\theta$ become enforced for θ in $[1, |O'_{\hat{i}}| + |O''_{\hat{i}}|)$. The situation is depicted in the figure to the left. Instead of thinking about the problem as an assignment problem, the result follows by viewing this as a network flow problem (similar perspectives are found in [8] and [10]), where each sailor has a demand of one unit and each job is a source with a capacity of one unit. First, the flow across the arcs from $O'_{\hat{i}}$ into $\hat{i}$ that do not already have a value of 1 can increase until the flow into $\hat{i}$ is $|O'_{\hat{i}}|$. Second, the flow along the arcs from $O''_{\hat{i}}$ into $\hat{i}$ can increase to supply a flow of $|O''_{\hat{i}}|$. This follows because as more flow is directed into $\hat{i}$ away from $C'_{\hat{i}}$, the demand that each node in $C'_{\hat{i}}$ receive at least one unit may be satisfied by the jobs in $F_{\hat{i}}$. This means that so long as the flow into $\hat{i}$ is less than $|O'_{\hat{i}}| + |O''_{\hat{i}}|$, no new bounds are enforced, and hence, $(B(1)|N(1))$ is optimal so long as $\theta < |O'_{\hat{i}}| + |O''_{\hat{i}}|$. ■

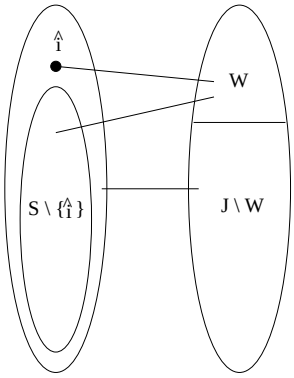
From Theorem 4 we have that $\theta^1_{\hat{i}}$ is loosely interpreted as the amount of competition that sailor $\hat{i}$ has for jobs. If $\theta^1_{\hat{i}}$ is large, we know that $O_{\hat{i}}$ is large, and hence, there are many jobs that sailor $\hat{i}$ can choose optimally. What we do not know from a large $\theta^1_{\hat{i}}$ is if the jobs in $O_{\hat{i}}$ are heavily contested by other sailors. Still, if $\theta^1_{\hat{i}}$ is larger than $\hat{M}_{\hat{i}}$, this indicates to the detailer that sailor $\hat{i}$ could choose any of the jobs in $O_{\hat{i}}$ with only minor degradation to subsequent lists. If $\theta^1_{\hat{i}}$ is small, we know that there are only a few (maybe none) free choices for sailor $\hat{i}$. As an example, suppose that $\theta^1_{\hat{i}}$ is 1. Then, we have that at best one of $O'_{\hat{i}}$ or $O''_{\hat{i}}$ contains a single job (we note that since $\theta^1_{\hat{i}}$ is at least 1, the bound in Theorem 4 always permits one of $O'_{\hat{i}}$ or $O''_{\hat{i}}$ to contain a job).

While Theorem 4 shows that $\theta^1_{\hat{i}}$ provides some insight into the competition that sailor $\hat{i}$ has for jobs, it makes no statement about how the optimal assignments for all sailors are effected as jobs are removed from the requisition list and put on sailor $\hat{i}$'s job list. The next result shows that $\theta^1_{\hat{i}}$ gauges this effect as well by establishing that $\theta^1_{\hat{i}} \geq K(G^*) + 1$. This follows because Corollary 1 says that so long as the collection of jobs removed from the requisition list

is smaller than $K(G^*)$, we are guaranteed that all sailors may be matched optimally to jobs on the diminished requisition list. In other words, $K(G^*)$ is the largest amount of arbitrary jobs that can be removed from the requisition list so that we are still guaranteed an optimal matching. In fact, from the definition of $K(G^*)$ we know that there is a collection of $K(G^*)+1$ jobs whose removal forces the existence of a group of sailors that can no longer be optimally matched to the jobs on the diminished requisition list. So, $K(G^*)$ measures how many jobs can be arbitrarily put on a sailor's job list so that an optimal matching for the rest of the sailors still exists. Since Theorem 5 shows that $\theta^1_{\hat{i}} \geq K(G^*)+1$, we have that a small $\theta^1_{\hat{i}}$ value indicates that there is a collection of sailors such that the number of optimal jobs for these sailors is roughly the same as the number of sailors. So, if $\theta^1_{\hat{i}}$ is 1, we are guaranteed that there is a collection of sailors that are matched to an equal number of optimal jobs, and hence, the competition among this group is high. In general, Theorem 5 guarantees that somewhere in G^* there is a collection of sailors whose optimal jobs outnumber the sailors by at most $\theta^1_{\hat{i}} - 1$. Sailor $\hat{i}$ may, or may not, be in this group of sailors, but the point is that a small $\theta^1_{\hat{i}}$ value indicates that such a group of sailors exists.

Theorem 5 *We have that* $\theta^1_{\hat{i}} \geq K(G^*)+1$.

Proof: Since $|\{\hat{i}\}| + K(G^*) \leq |O_{\hat{i}}|$, we know that sailor $\hat{i}$ has at least $K(G^*)+1$ optimal jobs in G^*. Let $W \subset O_{\hat{i}}$ be such that $|W| = K(G^*)$. From Corollary 1 we have that the subgraph $\hat{G}^* = (S, (J \backslash W), \mathcal{A}^* \backslash \{(i,j) \in \mathcal{A}^* : j \in W\})$ has the property that the elements in S may be matched to the elements in $J \backslash W$. In other words, the sailors can be matched optimally to jobs outside of W. As in the proof of Theorem 4, the result follows by considering our problem as a network flow problem over G^*, again with each sailor requiring one unit and each job having a capacity of one unit. As before, we show that the value of $x^*_{\mathcal{A}^*}$ may be adjusted so that no new bounds of $ASPR_\theta$ are forced for $\theta \in (1, K(G^*)+1)$. Since the sailors can be matched to the jobs in $J \backslash W$, we have that each sailor's required one unit can be accommodated by the jobs in $J \backslash W$. This means the flows from W into $S \backslash \{\hat{i}\}$ can decrease, which allows the flow from W into $\hat{i}$ to increase. So, no new bounds are enforced so long as the demand at $\hat{i}$ is less than $|W| + 1 = K(G^*) + 1$, where the added unit is contributed from the matching between S and $J \backslash W$. ■



The scope of Theorem 4 is different from the scope of Theorem 5. This is because Theorem 4 depends exclusively on the competition of a single sailor, and Theorem 5 depends on $K(G^*)$. Since $K(G^*)$ is independent of any particular sailor, we actually have that $\min\{\theta^1_i : i \in S\} \geq K(G^*)+1$. So, if $\theta^1_{\hat{i}} = 4$ and $\theta^1_{\bar{i}} = 1$, we know that some group of sailors, say S', is competing for a collection of optimal jobs that is the same size as S'. Moreover, we know that sailor $\hat{i}$ is not in this group. This is because $\theta^1_{\hat{i}}$ being 4 means that it is possible to assign 4 optimal jobs to sailor $\hat{i}$ and still guarantee that every other sailor has an optimal assignment. However, every sailor in S' can be assigned at most one optimal job if the other sailors in S' are to have an optimal assignment. So, a detailer knows that there is more leeway when forming sailor $\hat{i}$'s job list.

Both of the techniques developed below calculate the break points of $f^*_{\hat{i}}$, and we report the value of $\theta^1_{\hat{i}}$ as a level of competition for sailor $\hat{i}$. We also have the option of calculating

Algorithm 2(a)

Step 1: Solve $ASPR_1$ with a (PFIP) algorithm and calculate $\mathcal{A}^*(1)$. Set $k = 1$, and $t = 2$.

Step 2: Let $\mathcal{L} = \mathcal{J}_{\hat{i}}^0 = \{j : (\hat{i}, j) \in \mathcal{A}^*(1)\}$

Step 3: If $|\mathcal{L}| \geq \hat{M}_{\hat{i}}$, stop.

Step 4: Apply Algorithm 1 to find $\mathcal{A}^*(t)$ and $\theta_{\hat{i}}^k$.

Step 5: In order, set $\mathcal{J}_{\hat{i}}^k = \{j : (\hat{i}, j) \in \mathcal{A}^*(t)\} \backslash \mathcal{L}$ and $\mathcal{L} = \mathcal{L} \cup \mathcal{J}_{\hat{i}}^k$.

Step 6: If $|\mathcal{L}| \geq \hat{M}_{\hat{i}}$, stop.

Step 7: If $\theta_{\hat{i}}^k = t$, in order set $t = \theta^k + 1$, $k = k + 1$, and go to step Step 4.

Step 8: Set $\mathcal{J}_{\hat{i}}^{k+1} = \{j : (\hat{i}, j) \in \mathcal{A}^*(\theta_{\hat{i}}^k)\} \backslash \mathcal{L}$ and then set $\mathcal{L} = \mathcal{L} \cup \mathcal{J}_{\hat{i}}^{k+1}$.

Step 9: If $|\mathcal{L}| \geq \hat{M}_{\hat{i}}$, stop.

Step 10: In order, set $t = \theta_{\hat{i}}^k + 1$, $k = k + 2$ and go to Step 4.

Table 1: An algorithm that generates a job list $\mathcal{L}$ for sailor $\hat{i}$. The job list is partitioned by $B_{\hat{i}}^0, B_{\hat{i}}^1, \ldots, B_{\hat{i}}^k$.

$\min\{\theta_i^1 : i \in S\}$ and reporting this value as a measure of worst case competition. However, this requires that p linear programs be solved, and we typically do not undertake this calculation (it is not always necessary to calculate every $\theta_{\hat{i}}^1$, since as soon as one of these values is 1 we know that $K(G^*)$ is 0). We do not claim that $\theta_{\hat{i}}^1$ is a perfect measure of competition, but the discrepancy between $\theta_{\hat{i}}^1$ and $\hat{M}_{\hat{i}}$ does provide a rough estimate about the suboptimality that is required to achieve a satisfactory list length. As an example, suppose that sailor $\hat{i}$ has performed well and is near the end of her/his contract. Also, suppose that the guidance list length is $\hat{M}_{\hat{i}} = 5$ and that $\theta_{\hat{i}}^1$ is 1. This means there is at best one free job choice for sailor $\hat{i}$, and that any of the other jobs that could be optimally assigned are being contested by other sailors (it also indicates that sailor $\hat{i}$ might be competing for the most heavily contested jobs). So, to reach a list length of 5 we are most likely going to remove 4 jobs from consideration for the next few sailors. So, there is at most one 'good' job for sailor $\hat{i}$ (relative to the rest of the optimal assignments), and the detailer may want to explain the situation, offer the one job if it exists, and ask the sailor to call back when a new requisition list is available.

We investigate two techniques to generate a job list, denoted by $\mathcal{L}$. Each list is a partitioned and ordered set, and we use the notation $\mathcal{L} = (j_1, j_2 | j_3 | j_4, j_5)$ to indicate that the list contains jobs j_1 through j_5, but that the first two jobs are equally 'best' candidates, job j_3 is the second best candidate, and jobs j_4 and j_5 are equally best third candidates. We also assume that jobs unioned to the list are added to the end of the list. Both of the techniques use Algorithm 1 to calculate optimal partitions, and both segment the job list into groups of jobs that are 'best' assignments, 'next best' assignments, and so forth. The techniques differ in how they use the information provided by the optimal partition. The first of these techniques is Algorithm 2(a) and is found in Table 1. The algorithm begins by solving the *LP* relaxation of *AS* and locating all jobs that sailor $\hat{i}$ can be assigned to optimally. If there are enough of these jobs, the algorithm terminates, but if the list is not of sufficient length, Algorithm 1 is used to increase

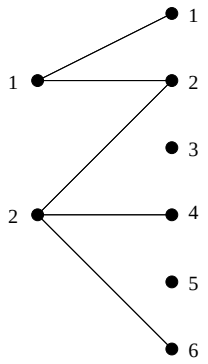


Figure 3: The optimal solutions for $\theta = 1$. We have that $\mathcal{J}_1^0 = \{1, 2\}$, and the current list is $\mathcal{L} = (1, 2)$.

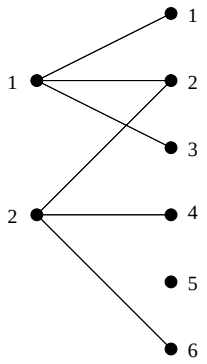


Figure 4: The optimal solutions for $\theta = 2$. Since job 3 is now optimal for sailor 1, we have that $\mathcal{J}_1^1 = \{3\}$. The list is now $\mathcal{L} = (1, 2|3)$.

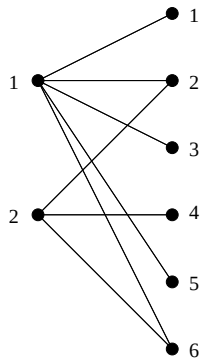


Figure 5: The optimal solutions for $\theta = 3$. Jobs 5 and 6 are now optimal for sailor 1, and $\mathcal{J}_1^2 = \{5, 6\}$. The final list is $\mathcal{L} = (1, 2|3|5, 6)$

θ to the next break point, $\theta_{\hat{i}}^k$. There are two situations to consider. First, if $\theta_{\hat{i}}^k$ is equal to the next integer value of θ, represented by t in the algorithm, we have that the partition at the next break point is the partition that we are looking for. In this case, Step 5 collects any newly found optimal jobs in $\mathcal{J}_{\hat{i}}^k$ and adds these jobs to the list. Remember that these jobs are optimal relative to the new value of θ, and since all possible optimal assignments are listed in $\mathcal{J}_{\hat{i}}^0$, we have that any jobs added to $\mathcal{L}$ in Step 5 are not optimal to the original AS problem. So, if sailor $\hat{i}$ chooses a job in $\mathcal{J}_{\hat{i}}^1$, we know that this choice is suboptimal in the original AS problem. However, this is the price we pay to have a sufficient list length. This is not to say that the jobs in $\mathcal{J}_{\hat{i}}^1$ are not optimal in any sense, for they are optimal in the situation that we want to increase sailor $\hat{i}$'s minimum list length past 1, while at the same time require that the other sailors be assigned at least one job. The algorithm stops at this point if the list length is at least the guidance value, and we point out that this termination criteria depends on the length of $\mathcal{L}$ and not directly on θ. Steps 8, 9, and 10 are needed if $\theta_{\hat{i}}^k$ is greater than the next integer value of θ, for if $\theta_{\hat{i}}^k > t$, we have that $(B(t)|N(t))$ is different from $(B(\theta_{\hat{i}}^k)|N(\theta_{\hat{i}}^k))$. Consequently, we have that $\mathcal{A}^*(t)$ is not necessarily the same as $\mathcal{A}^*(\theta_{\hat{i}}^k)$. For example, if $\theta_{\hat{i}}^1$ is 3, we know that $(B(2)|N(2))$ remains optimal so long as θ is in $[1, 3)$. However, when θ attains the value of 3 we get the new optimal partition $(B(3)|N(3))$. In this situation, Step 8 adds to the list any jobs in $\mathcal{A}^*(\theta_{\hat{i}}^k)$ that are not already on the list. If the list is sufficiently long, the algorithm terminates, but if not, we update the algorithm parameters and continue.

Observe that θ increases as little as possible to guarantee a sufficient list length. An example of how the process works is illustrated in Figures 3, 4, and 5. We assume in this example that we are generating a list for sailor 1 and that the guidance value for the list length is $\hat{M}_{\hat{i}} = 4$. An arc is shown if the assignment can be made optimally. We point out that θ_1^1 for this problem is 2, which implies that up to 2 of the 4 jobs to go on the list are free choices. In this example, jobs 1 and 2 are indeed free choices and are placed on sailor 1's list.

The next algorithm is, in some sense, a refinement of Algorithm 2(a), by which we mean that the segmentation of the job list is less course. While the interpretation of the second procedure is significantly different, there are only two procedural differences between Algorithms 2(a) and

Algorithm 2(b)

Step 1: Solve $ASPR_1$ with a (PFIP) algorithm and calculate $\mathcal{A}^*(1)$. Set $k = 1$, and $t = 2$.

Step 2: Let $\mathcal{L} = O^!_{\hat{i}}$

Step 3: If $|\mathcal{L}| \geq \hat{M}_{\hat{i}}$, stop.

Step 4: Apply Algorithm 1 to find $\mathcal{A}^*(t)$ and $\theta^k_{\hat{i}}$.

Step 5: Set $\mathcal{J}^k_{\hat{i}} = \{j : (\hat{i}, j) \in \mathcal{A}^*(t) \cap \mathcal{A}^*(1)\} \backslash \mathcal{L}$ and $\mathcal{J}^{k+1}_{\hat{i}} = \{j : (\hat{i}, j) \in \mathcal{A}^*(t)\} \backslash (\mathcal{J}^k_{\hat{i}} \cup \mathcal{L})$

Step 6: Set $\mathcal{L} = \mathcal{L} \cup \mathcal{J}^k_{\hat{i}} \cup \mathcal{J}^{k+1}_{\hat{i}}$.

Step 7: If $|\mathcal{L}| \geq \hat{M}_{\hat{i}}$, stop.

Step 8: If $\theta^k_{\hat{i}} = t$, in order set $t = \theta^k + 1$, $k = k + 2$, and go to step Step 4.

Step 9: Set $\mathcal{J}^{k+2}_{\hat{i}} = \{j : (\hat{i}, j) \in \mathcal{A}^*(\theta^k_{\hat{i}}) \cap \mathcal{A}^*(1)\} \backslash \mathcal{L}$ and $\mathcal{J}^{k+3}_{\hat{i}} = \{j : (\hat{i}, j) \in \mathcal{A}^*(\theta^k_{\hat{i}})\} \backslash (\mathcal{J}^{k+2}_{\hat{i}} \cup \mathcal{L})$

Step 10: Set $\mathcal{L} = \mathcal{L} \cup \mathcal{J}^{k+2}_{\hat{i}} \cup \mathcal{J}^{k+3}_{\hat{i}}$.

Step 11: If $|\mathcal{L}| \geq \hat{M}_{\hat{i}}$, stop.

Step 12: In order, set $t = \theta^k_{\hat{i}} + 1$, $k = k + 4$ and go to Step 4.

Table 2: A second algorithm to generate a job list. Notice that each iteration adds two sets to the list.

Algorithm 2(b). First, $\mathcal{J}^0_{\hat{i}}$ no longer contains all the jobs that are optimal for sailor $\hat{i}$ in the original *AS* problem. Instead, only the jobs for which sailor $\hat{i}$ is exclusively optimal are included in $\mathcal{J}^0_{\hat{i}}$. So, because job 2 is not exclusively optimal for sailor 1 in Figure 3, we have that $\mathcal{J}^0_1$ is $\{1\}$ instead of $\{1, 2\}$. Second, at every iteration of Algorithm 2(b) we segment the newly added jobs into those that are optimal in the original *AS* problem, and those that are not. The reason for this refinement is that there are instances where a job is optimal for $\theta = 1$ but not for larger values of θ. This means that optimal jobs in the original *AS* problem may become suboptimal as the minimum list size increases. This is possible because sailor $\hat{i}$ is competing with other sailors for optimal jobs. One way to view Algorithm 2(b) is that it uses the optimal solutions to the original *AS* problem as a master plan. As θ increases from 1, the competition for jobs alters the optimal assignments. The jobs that match the master plan appear higher on the list than those that do not. As an example, consider the list generated by Algorithm 2(b) for $\hat{M}_1 = 3$ and the situation depicted in Figures 3, 4, and 5. The optimal assignments in Figure 3 form the master plan, meaning that we should attempt to match these assignments as much as possible. As already mentioned, the algorithm begins by setting $\mathcal{J}^0_1$ to $\{1\}$. For $\theta = 2$ we have that jobs 2 and 3 are optimal, but these jobs are segmented into $\mathcal{J}^1_1 = \{2\}$ and $\mathcal{J}^2_1 = \{3\}$. This segmentation is imposed because job 2 was optimal for sailor 1 in the master plan, but job 3 was not. So, in this case the list is the same, but the segmentation is refined —i.e. $\mathcal{L}$ is $(1|2|3)$ instead of $(1, 2|3)$. The difference is that the list generated by Algorithm 2(a) indicates an indifference as to whether or not sailor 1 chooses job 1 or 2, but the list generated by Algorithm 2(b) clearly demonstrates a preference to assigning sailor 1 to job 1.

The example illustrated in Figures 6, 7, and 8 shows that the lists generated by the Algorithms can differ more substantially. We assume that the guidance list length is $\hat{M}_{\hat{i}} = 4$, and

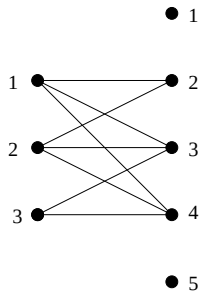


Figure 6: The optimal solutions for $\theta = 1$ form the master plan. Since no jobs are uniquely optimal for sailor 1, $\mathcal{J}_{\hat{i}}^0 = \emptyset$.

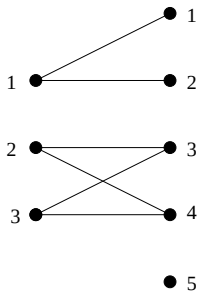


Figure 7: The optimal solutions for $\theta = 2$. Since job 2 is optimal for sailor 1 here and in the master plan, $\mathcal{J}_{\hat{i}}^1 = \{2\}$. Since job 1 is not optimal in the master plan but is optimal here, $\mathcal{J}_{\hat{i}}^2 = \{1\}$. The current list is $\mathcal{L} = (2|1)$.

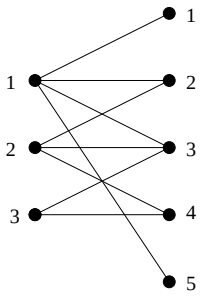


Figure 8: The optimal solutions for $\theta = 3$. Jobs 3 and 5 are now optimal for sailor 1. Since job 3 is in the master plan and job 5 is not, we have that $\mathcal{J}_{\hat{i}}^3 = \{3\}$ and $\mathcal{J}_{\hat{i}}^4 = \{5\}$. The final list is $\mathcal{L} = (2|1|3|5)$.

we point out that θ_1^1 is 1. We have from Theorem 4 that there is at most one fee job for sailor 1, and from Theorem 5 we have that $K(G^*) = 0$, which follows because the set of all three sailors is optimally matched to exactly 3 jobs. Notice that θ_1^1 is only 1/4 of the desired jobs, whereas this ratio was 1/2 in the previous example. The list produced by Algorithm 2(a) is $\mathcal{L} = (2, 3, 4|1)$. However, the list generated by Algorithm 2(b) is $\mathcal{L} = (2|1|3|5)$. Notice that not only is the order different, but that the jobs themselves are different. The list generated by Algorithm 2(a) indicates an indifference as to whether or not sailor $\hat{i}$ receives jobs 2, 3, and 4 as choices. This is because sailor $\hat{i}$ can choose any of these jobs and still guarantee that future sailors receive an optimal choice. However, offering jobs 2, 3, and 4 to sailor $\hat{i}$ takes them off the market, and subsequent sailors must wait until sailor $\hat{i}$ makes a choice before they can consider the unchosen jobs. Algorithm 2(b) takes this into consideration, and the interpretation of the list generated by this algorithm is different. The first segment contains job 2 and indicates that if a single job is to be (temporarily) removed from subsequent consideration, this is the job sailor 1 should receive (if there were more jobs in this segment, sailor 1 would receive these as first choices as well). The second segment contains job 1 and informs a detailer that if two jobs are to be (temporarily) removed from the requisition list, jobs 2 and 1 are the optimal jobs to offer sailor 1. The meaning of the third and fourth segments is similar. If the detailer is going to remove 3 or 4 jobs from the requisition list, the jobs to offer sailor 1 are respectively (2|1|3) and (2|1|3|5). The reason that job 4 does not appear on this list is because it never reappears as an optimal choice as multiple jobs are removed from the requisition list.

Notice that Algorithm 2(a) tends to favor the sailor that is asking for a job list, and that Algorithm 2(b) tends to 'save' jobs for subsequent sailors. This is clear in the last example where job 4 is an optimal assignment for sailor 1. In Algorithm 2(a) job 4 is one of the 'best' job candidates, but in Algorithm 2(b) job 4 does not appear on sailor 1's list because it is an important job choice for either sailor 2 or 3. Because we do not know which sailors are going to contact their detailer within the planning period, the extra hedging provided by Algorithm 2(b) may not be warranted. Both techniques have been prototyped in Matlab, and a topic of current research is to use historical data to see which of these techniques is most appropriate.

5 Conclusions

The goals of this work were threefold. Our first goal was to show that there was value in having a strictly complementary solution to a relaxed binary problem, and Theorem 1 clearly demonstrates that this is the case. The second goal was to show that continuous parametric techniques are capable of providing useful information about an assignment problem. Theorems 4 and 5 show that the first break point of the objective value provides an insight into the structure of the optimal assignments. Both of the above goals were inspired by the need to develop techniques that supported the complicated assignment process undertaken by Navy detailers, and our third goal was to develop helpful techniques. Algorithms 2(a) and 2(b) show that the optimal partition provides meaningful information about this assignment process. These two techniques are currently being tested with historical data to see which one more appropriately satisfies the needs of the Navy.

Acknowledgments

The author would like to thank the members of the Hearin Center for Enterprise Science for their comments on earlier revisions. Also, this work would not have been possible without the aid of several people at the Naval Personal Research Science and Technology (NPRST) department at the Millington Naval Base.

References

[1] A. Ali, J. Kennington, and T. Liang. Assignment with en route training of navy personnel. *Navel Research Logistics*, 40:581–592, 1993.

[2] T. Blanco and R. Hillery. A sea story: Implementing the navy's personnel assignment system. *Operations Research*, 42(5):814–822, 1994.

[3] T. Gal. *Postoptimal analysis, parametric programming, and related topics: degeneracy, multicrtiteria decision making, redundancy.* Walter de Gruyter & Co., New York, NY, 2nd edition, 1995.

[4] A. Goldman and A. Tucker. Theory of linear programming. In H. Kuhn and A. Tucker, editors, *Linear Inequalities and Related Systems*, volume 38, pages 53–97. Princeton University Press, Princeton, New Jersey, 1956.

[5] H. Greenberg. *Mathematical Programming Glossary.* World Wide Web, `http://www-math.cudenver.edu/~hgreenbe/glossary/glossary.html`, 1996-2001.

[6] P. Hall. On representation of subsets. *Journal of London Mathematical Society*, 10:26–30, 1935.

[7] A. Holder. Partitioning multiple objective solutions with applications in radiotherapy design. Technical Report 54, Trinity University Mathematics, 2001.

[8] D. Klingman and N. Phillips. Topological and computational aspects of preemptive multicriteria military personnel assignment problems. *Management Science*, 30(11):1362–1375, 1984.

[9] T. Liang and B. Buclatin. Improving the utilization of training resources through optimal personnel assignment in the u.s. navy. *European Journal of Operational Research*, 33:183–190, 1988.

[10] T. Liang and T. Thompson. A large-scale personnel assignment model for the navy. *Decision Sciences*, 18(2):235–250, 1987.

[11] L. McLinden. An analogue of Moreau's approximation theorem, with applications to the nonlinear complementarity problem. *Pacific Journal of Mathematics*, 88(1):101–161, 1980.

[12] L. McLinden. The complementarity problem for maximal monotone multifunctions. In R. Cottle, R. Giannessi, and J. Lions, editors, *Variational Inequalities and Complementarity Probelems*, pages 251–270. John Wiley & Sons, 1980.

[13] R. Monteiro and S. Mehrotra. A general parametric analysis approach and its implication to sensitivity analysis in interior point methods. *Mathematical Programming*, 72:65–82, 1996.

[14] C. Roos, T. Terlaky, and J.-Ph. Vial. *Theory and Algorithms for Linear Optimization: An Interior Point Approach.* John Wiley & Sons, New York, NY, 1997.

[15] S. Wright. *Primal-Dual Interior-Point Methods.* SIAM, Philadelphia, PA, 1997.

[16] Y. Ye. *Interior Point Algorithms Theory and Analysis.* John Wiley & Sons, Inc., New York, NY, 1997.