On the Solution to the Nonlinear Resource Allocation Problem Using Variable Fixing and Interior-Point Methods

James Rohal

Advisor: Dr. Stephen Wright

Contents

List of Figures

List of Algorithms

1 Introduction

We will be studying the nonlinear resource allocation problem (P):

$$\begin{aligned} (P) \quad \min \quad & f(\xi) := \sum_{i=1}^{n} f_i(\xi_i) \\ \text{st} \quad & g(\xi) := \sum_{i=1}^{n} g_i(\xi_i) \leqslant M \\ & \ell_i \leqslant \xi_i \leqslant u_i, \end{aligned}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}^n$ and $g\colon \mathbb{R}^n \to \mathbb{R}^n$ are separable, convex, differentiable functions and ℓ_i and u_i are finite bounds on our decision variables. We assume that the feasible region is nonempty and bounded and that each f_i and g_i are monotone. In practice, we may replace the inequality constraint by an equality constraint. This is motivated by the idea that we expect the explicit constraint to be active at an optimal solution [7]. In many applications, the decision variables are integral, but we only concern ourselves with the continuous relaxation where the decision variables can be real valued. The techniques we describe below can be extended to use methods like branch and bound, dynamic programming, or converting to a 0,1 knapsack problem, each of which solve the integer version of our problem

Since this problem is applicable to many fields such as economics, engineering, statistics, and manufacturing, researchers have studied the problem extensively and published hundreds of articles with the first being in 1953 [6]. A survey paper by Michael Patriksson [7] attempts to give a broad view of work done on problem (P) by describing its history and applications and describing the most common techniques used to solve it. Among these techniques include Lagrange multiplier methods and pegging methods which we discuss in sections and 2.1 and 2.2. The applications we will be looking at come from the fields of statistics, manufacturing, and quadratic programming. By making use of the pegging methods of Bretthauer and Shetty [2] and using a relatively new set of methods called interior-point methods, we have created a way to solve (P) quickly that is competitive with many of the methods mentioned in Patriksson's paper.

In the next section we describe three methods that were previously used to solve (P) and the theory behind them. The third chapter concerns itself with indicators, which is a class of functions used for identifying active variables. The implementation of the various algorithms using MATLAB are discussed in the last section along with a description of a new method that is competitive (in speed) with previous methods. We also present our numerical results for the various methods and mention future work that can be done.

Throughout this paper we use the convention that a superscript * denotes an optimal value and **bold symbols** refer to vectors.

2 Methods

2.1 Lagrange Multiplier Methods

Among the oldest of the methods used to solve problem (P), Lagrange multiplier methods find the optimal value of the Lagrange multiplier $\rho \geqslant 0$ associated with the explicit constraint. We obtain the following optimality conditions an optimal value ξ^* must satisfy [7]:

$$\begin{aligned} \nabla f(\xi^*) + \rho^* \nabla g(\xi^*) &= 0 \\ g(\xi^*) - M &\leqslant 0 \\ \ell_i \leqslant \xi^* &\leqslant u_i \\ \rho^*(g(\xi^*) - M) &= 0 \\ \rho^* &\geqslant 0. \end{aligned}$$

Furthermore ξ^* must satisfy the tangent criterion

$$\begin{aligned} \xi_i^* = \ell_i, \quad & \text{if } f_i'(\xi_i^*) \geqslant -\rho^* g_i'(\xi_i^*) \\ \xi_i^* = u_i, \quad & \text{if } f_i'(\xi_i^*) \leqslant -\rho^* g_i'(\xi_i^*) \\ \ell_i < \xi_i^* < u_i, \quad & \text{if } f_i'(\xi_i^*) = -\rho^* g_i'(\xi_i^*). \end{aligned} \tag{2.1}$$

The above conditions are the optimality conditions for the minimization of the concave piecewise linear Lagrange dual function of a single variable ρ,

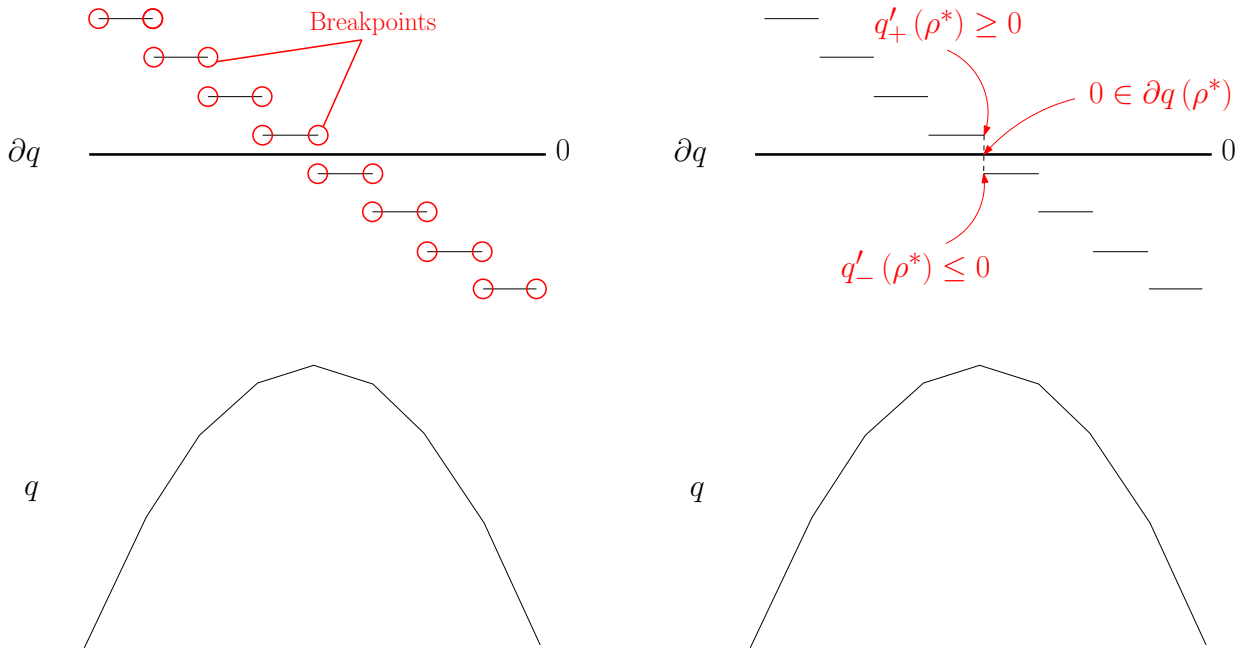
$$q(\rho) := -M\rho + \sum \min_{\ell_i \leqslant \xi_i \leqslant u_i} \left\{ f_i(\xi_i) + \rho g_i(\xi_i) \right\}.$$

There is no guarantee that the derivative q' exists, so to find an optimal value ρ^* one can use a bisection search or a breakpoint search. A breakpoint is a value for ρ that satisfies the tangent criterion 2.1 which can be seen in Figure 1(a).

A rudimentary breakpoint search method first places the breakpoints in some list $\{\rho_1, \ldots, \rho_N\}$ (where $N \leqslant 2n$ since there may be times when ρ satisfies more than one condition of 2.1). Finding ρ^* then amounts to finding an index i^* where $q'(\rho_{i^*}) = 0$ (hence we are done) or two indices j and k where $q'_+(\rho_j) > 0$ and $q'_-(\rho_k) < 0$ and then performing an interpolation between these two values so that $\rho^* \in (\rho_j, \rho_k)$ and $0 \in \partial q(\rho^*)$. This can be seen in Figure 1(b).

It is possible to speed up the search for this optimal breakpoint by first sorting the list of breakpoints. Although the sorting operation usually takes $O(n \log n)$ time, using a median search takes $O(n)$ time and can significantly speed up the process of finding ρ^*.

Although we didn't explicitly test breakpoint methods during this project, it should be considered in any future work on this subject.



(a) Breakpoints labeled on the subdifferential ∂q.

(b) An optimal ρ^* will satisfy the condition that $0 \in \partial q(\rho^*)$.

Figure 1: *Illustration of the breakpoint search method.*

2.2 Pegging Methods

Pegging methods find an optimal solution by solving relaxations of (P) where the finite bound constraints on ξ are removed. It is a recursive algorithm because at each stage some variables receive their optimal value and are thus removed from the problem, creating a smaller subproblem to solve. The original pegging method was motivated by work done by Bitran and Hax [1] and Robinson et al. [8] and described succinctly in the paper by Bretthauer and Shetty [2]. In our computational tests, we found that the pegging method is among the fastest ways of solving the problem at hand as long as a closed-form solution can be found to the subproblem.

First, let ρ denote the Lagrange multiplier for $\sum_{i=1}^{n} g_i(\xi_i) \leqslant M$. We assume that the optimal value for ρ is positive, otherwise we could check for an optimal solution quickly. Furthermore, we assume that a solution $\overline{\xi}_i(\rho)$ exists to the nonlinear equation $f'_i + \rho g'_i = 0$ as a function of ρ. This gives us two major cases to consider.

Case 1: $g_i(\xi_i)$ decreasing in ξ_i for all i and $\overline{\xi}_i(\rho)$ is increasing in ρ for all i.

Case 2: $g_i(\xi_i)$ increasing in ξ_i for all i and $\overline{\xi}_i(\rho)$ is decreasing in ρ for all i.

Let ξ^* denote the optimal solution to (P). We refer to a variable as being pegged if ξ^*_i is fixed at its upper or lower bound, u_i or ℓ_i, respectively. Let L^k denote the index set of variables pegged to their lower bound and U^k denote the index set of variables pegged to their upper bound up to iteration k of the pegging algorithm. We let I^k denote the index

set of unpegged variables at iteration k. The pegging algorithm for solving (P) when Case 1 is satisfied is as follows [2].

Algorithm 1: *Pegging Algorithm for Problems Satisfying Case 1*

Set $I^1 = \{1, \ldots, n\}$, $L^1 = \emptyset$, $U^1 = \emptyset$.
For $k = 1$ **to** n
Solve the current subproblem (P^k) and let ξ_i^k for all $i \in I^k$ denote its optimal solution where

$$(P^k) \quad \min \quad \sum_{i \in I^k} f_i(\xi_i) + \sum_{i \in L^k} f_i(\ell_i) + \sum_{i \in U^k} f_i(u_i)$$
$$\text{st} \quad \sum_{i \in I^k} g_i(\xi_i) + \sum_{i \in L^k} g_i(\ell_i) + \sum_{i \in U^k} g_i(u_i) = M.$$

Identify the variables in the subproblem solution that do not satisfy their bounds:

$$S^k = \left\{ i \in I^k : x_i^k < \ell_i \right\} \qquad \text{and} \qquad B^k = \left\{ i \in I^k : x_i^k > u_i \right\}.$$

If $S^k \cup B^k = \emptyset$ **then End For**.
Calculate total lower and upper infeasbilities:

$$\nabla = \sum_{i \in S^k} \left[g_i(\ell_i) - g_i\left(\xi_i^k\right) \right] \qquad \text{and} \qquad \Delta = \sum_{i \in B^k} \left[g_i\left(\xi_i^k\right) - g_i(u_i) \right].$$

If $\nabla \geqslant \Delta$ **then**
Set $I^{k+1} = I^k \setminus B^k$, $U^{k+1} = U^k + B^k$, $L^{k+1} = L^k$.
Else
Set $I^{k+1} = I^k \setminus S^k$, $L^{k+1} = L^k + S^k$, $U^{k+1} = U^k$.
End If
End For
Return optimal solution:

$$\xi_i^* = \begin{cases} \ell_i & \text{if } i \in L^k \\ u_i & \text{if } i \in U^k \\ \xi_i^k & \text{if } i \in I^k. \end{cases}$$

Since at least one variable is pegged at each iteration, the method terminates in a finite number of steps with a complexity at most $O(n^2)$. The speed of this algorithm relies on whether we can find a closed form solution for (P^k) in terms of the multiplier ρ. We show in our numerical tests that when a basic numerical solver is used, then the pegging method is quite slow. A natural extension of this algorithm would be a method to give a better determination of the pegged variables. We use the idea of indicators in section 3 to accomplish this.

2.3 Interior-Point Methods

Interior-point methods are iterative methods that generate a series of iterates that lie in the interior of the feasible region. It is possible to avoid the boundary by introducing barrier functions that create a "penalty" for getting too close. Primal-dual interior-point methods (of which we are concerned with in this paper) apply Newton methods to the optimality conditions so that we satisfy a strict inequality at each iteration. To do so, each iteration must modify the search directions and step lengths so that our within a reasonable neighborhood of the central path. The central path (whose existence is proved in [10]) is an arc of strictly feasible points along which we take steps to get closer to a primal-dual solution.

When working with interior-point methods, it is sometimes difficult to find a strictly feasible initial solution to start the method. To avoid this difficulty one could embed (P) into a larger program for which such a solution exists or modify the interior-point algorithm to not require a strictly feasible starting point [10]. Another difficulty with interior-point methods is that fast convergence requires problem specific tuning. For example, using a method like `fmincon` in MATLAB converges extremely slowly since it has not been optimized to solve problem (P).

2.3.1 Optimality Conditions

The benefit of working with the problem (P) is its nice structure. Recall that we have made the assumption that our explicit constraint holds as equality. We begin by introducing slack variables $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{s} \in \mathbb{R}^n$ for the upper and lower bound, respectively, to form a nonnegative system (P') which is equivalent to (P).

$$\begin{aligned} (P') \quad \min \quad & \sum_{i=1}^{n} f_i(x_i + \ell_i) \\ \text{st} \quad & \sum_{i=1}^{n} g_i(x_i + \ell_i) = M \\ & u_i - \ell_i = s_i + x_i \\ & x_i \geqslant 0, s_i \geqslant 0. \end{aligned}$$

The optimality (Karush-Kuhn-Tucker) conditions for (P') are as follows:

$$\nabla f(\mathbf{x} + \boldsymbol{\ell}) + \rho \nabla g(\mathbf{x} + \boldsymbol{\ell}) - \boldsymbol{\lambda} + \boldsymbol{\mu} = \mathbf{0} \tag{2.2}$$

$$\sum_{i=1}^{n} g_i(x_i + \ell_i) = M \tag{2.3}$$

$$u_i - \ell_i = s_i + x_i \tag{2.4}$$

$$\lambda_i x_i = 0 \tag{2.5}$$

$$\mu_i s_i = 0 \tag{2.6}$$

$$x_i \geqslant 0, s_i \geqslant 0, \mu_i \geqslant 0, \lambda_i \geqslant 0 \tag{2.7}$$

where λ and μ are the Lagrange multipliers for the lower and upper bound, respectively. The interior-point method we use is a path-following algorithm which perturbs the complementary slackness conditions. We thus modify the conditions (2.5) - (2.7) above to reflect this:

$$\nabla f(\mathbf{x}+\boldsymbol{\ell}) + \rho\nabla g(\mathbf{x}+\boldsymbol{\ell}) - \boldsymbol{\lambda} + \boldsymbol{\mu} = \mathbf{0} \tag{2.8}$$

$$\sum_{i=1}^{n} g_i(x_i+\ell_i) = M \tag{2.9}$$

$$u_i - \ell_i = s_i + x_i \tag{2.10}$$

$$\lambda_i x_i = \tau \tag{2.11}$$

$$\mu_i s_i = \tau \tag{2.12}$$

$$x_i > 0, s_i > 0, \mu_i > 0, \lambda_i > 0. \tag{2.13}$$

We can then define the central path as an arc of strictly feasible points where the complementary slackness conditions have the same value (τ) for all indices. An interior-point method then travels along the central path and reduces $\tau \downarrow 0$. The central path is beneficial since it guides the iterative procedure along a path avoiding spurious solutions and reduces the complementary slackness conditions to zero at a steady rate.

2.3.2 Interior-Point Algorithm

The equations from the KKT conditions (2.8) - (2.13) form a nearly linear system, so it is reasonable to attempt to solve (P) using a Newton-like algorithm (and exploit the structure to improve efficiency). We begin by restating the optimality conditions in the form:

$$F(\mathbf{x},\boldsymbol{\lambda},\mathbf{s},\boldsymbol{\mu},\rho) = \begin{bmatrix} \nabla f(\mathbf{x}+\boldsymbol{\ell}) + \rho\nabla g(\mathbf{x}+\boldsymbol{\ell}) - \boldsymbol{\lambda} + \boldsymbol{\mu} \\ \boldsymbol{\lambda}\cdot\mathbf{x} - \tau \\ \boldsymbol{\mu}\cdot\mathbf{s} - \tau \\ \mathbf{x}+\mathbf{s}+\boldsymbol{\ell}-\mathbf{u} \\ \sum_{i=1}^{n} g_i(x_i-\ell_i) - M \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2.14}$$

A pure Newton step has a tendency to take us to a point where the positivity condition $(\mathbf{x},\boldsymbol{\lambda},\mathbf{s},\boldsymbol{\mu},\rho) > 0$ fails to hold. By biasing the search direction to point towards the non-negative orthant we can take longer steps while still being a feasible point. We introduce a centering parameter $\sigma \in [0,1]$ and a simple duality measure β defined by

$$\beta = \frac{\mathbf{x}^T\boldsymbol{\lambda} - \boldsymbol{\mu}^T\mathbf{s}}{n}.$$

If we wish to improve centrality; that is, move closer to the central path, we can let $\sigma = 1$ in which case we do not move much closer to a solution, but instead setup for a large step for the next iteration. On the other extreme, we can let $\sigma = 0$ which would give us a pure Newton step which is sometimes described as the affine-scaling direction. In practice, it is best to let $\sigma \in (0,1)$ vary during the iterations.

Replacing τ by the product $\sigma\beta$ in (2.14), then the Newton system we need to solve is

$$JF(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}, \boldsymbol{\mu}, \rho) \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\lambda} \\ \Delta\mathbf{s} \\ \Delta\boldsymbol{\mu} \\ \Delta\rho \end{bmatrix} = -F(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}, \boldsymbol{\mu}, \rho)$$

where $JF(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}, \boldsymbol{\mu}, \rho)$ is the Jacobian of $F(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}, \boldsymbol{\mu}, \rho)$.

Interior-point algorithms generate iterates $z^k = (\mathbf{x}^k, \boldsymbol{\lambda}^k, \mathbf{s}^k, \boldsymbol{\mu}^k, \rho^k)$ that give us strictly feasible points. To maintain strict feasibility, it is not always permissable to take a full Newton step, so instead we perform a line search in the Newton direction to find a new iterate

$$z^{k+1} = z^k + \alpha^k \Delta z^k$$

where $\alpha^k \in (0, 1]$ is a line search parameter. Since the parameter α^k tends to be small when we perform this line search, it is often necessary to modify Newton's method to allow for larger steps. We must also guarantee that the steps we take lead us to a solution. By staying close to the central path, the directions aim closer to the interior of the strictly feasible set which allows us to move further for each iteration. Furthermore, barrier functions are created to prevent the components of z^k from coming to close to the boundary of the feasible set, which would distort our solution and prevent our algorithm from progressing.

We now lay out a general framework for an interior-point method with iterates $z^k = (\mathbf{x}^k, \boldsymbol{\lambda}^k, \mathbf{s}^k, \boldsymbol{\mu}^k, \rho^k)$.

Algorithm 2: *Interior-Point Method Framework*

Find a strictly feasible point z^0.
For $k = 1, 2, \ldots$ **do**
  Solve the Newton system $JF\left(z^k\right) \Delta z^k = -F\left(z^k\right)$ where F is defined as in (2.14) .
  Set $z^{k+1} = z^k + \alpha^k \Delta z^k$ where α^k is chosen so $z^{k+1} > 0$.
End For

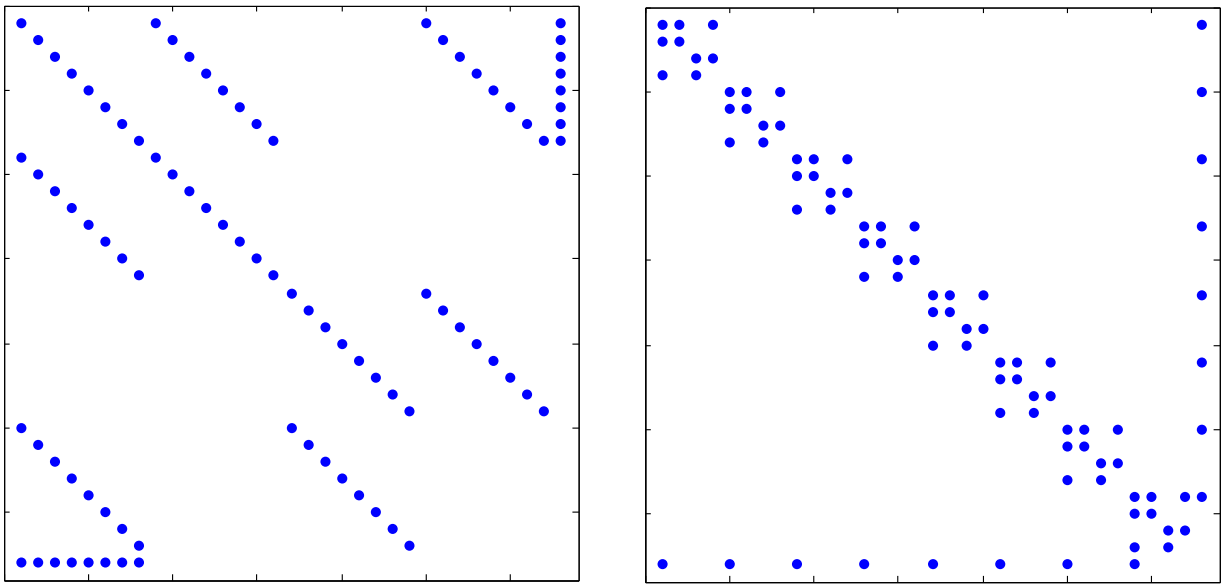
We can guarantee the interior-point method will terminate after a finite number of iterations by checking residuals or using a method of Ye that attempts to jump to an exact primal-dual solution [10].

One special feature of problem (P) is the structure of the Jacobian JF. We find

$$JF(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}, \boldsymbol{\mu}, \rho) = \begin{bmatrix} \operatorname{diag}\left(\frac{\partial^2 f}{\partial \xi_i^2} + \rho_i \frac{\partial^2 g}{\partial \xi_i^2}\right) & -I & 0 & I & \left(\frac{\partial g}{\partial \xi_i}\right)^T \\ \operatorname{diag}(\lambda_i) & \operatorname{diag}(x_i) & 0 & 0 & 0 \\ 0 & 0 & \operatorname{diag}(\mu_i) & \operatorname{diag}(s_i) & 0 \\ I & 0 & I & 0 & 0 \\ \left(\frac{\partial g}{\partial \xi_i}\right) & 0 & 0 & 0 & 0 \end{bmatrix}$$

where I is the $n \times n$ identity matrix and $\xi_i = x_i + \ell_i$. We may visualize the sparsity pattern of JF in Figure 2(a) and can permute the rows and columns to form a new matrix as seen

in Figure 2(b). The permuted matrix can be inverted using a Schur complement which is more efficient than finding the inverse of the unpermuted Jacobian.



(a) Unmodified JF. (b) JF with rows and columns permuted.

Figure 2: *Sparsity pattern of the Jacobian* JF. *Nonzero entries are colored blue while zero entries are white.*

3 Indicators

A technique for improving the speed of certain algorithms would be to identify active constraints early on in an iterative process. The problem of identifying active constraints is equivalent to asking which of the constraints $x_i \geqslant 0$ and $s_i \geqslant 0$ are active at ξ^*. We let

$$\mathcal{L}^* = \{i \colon x_i^* = 0, 1 \leqslant i \leqslant n\}$$
$$\mathcal{U}^* = \{i \colon s_i^* = 0, 1 \leqslant i \leqslant n\}$$

be the set of indices of variables that are zero at the solution ξ^* for the lower and upper bound, respectively. For our discussion below, we let $\mathcal{Z}^*$ denote any one of $\mathcal{L}^*$ or $\mathcal{U}^*$ exclusively.

By identifying these zero variables early on, we may form a smaller dimension subproblem to solve, providing a significant savings in computational work. Furthermore, removing these variables from the original problem may also improve the conditioning of the Jacobian [3].

3.1 Definition And Properties

Recall that $z^k = (x^k, \lambda^k, s^k, \mu^k, \rho^k)$ and the interior-point method is an iterative procedure of the form

$$z^{k+1} = z^k + \alpha^k \Delta z^k.$$

where α^k is the step length. Let $\left(z^k, \Delta z^k\right)$ be a sequence generated by the interior-point method. We define an indicator function I to be a function which assigns to $\left(z^k, \Delta z^k\right)$ an n-vector of extended reals $I\left(z^k, \Delta z^k\right)$ that satisfies the property that if $z^k \to z^*$, then for $i = 1, \ldots, n$

$$\lim_{k\to\infty} I_i^{\mathcal{Z}^*} = \begin{cases} 0, & \text{if } i \in \mathcal{Z}^* \\ \theta_i, & \text{if } i \notin \mathcal{Z}^* \end{cases}$$

where $\min_i \theta_i > 0$. It is desirable that any indicator $I^{\mathcal{Z}^*}$ have the following ideal properties [4]:

1. the sharp separation property,

$$\min \theta_i >> 0; \quad i \notin \mathcal{Z}^*$$

2. the uniform separation property,

$$\theta_i = \theta; \quad i \notin \mathcal{Z}^*$$

   for some nonzero constant θ,

3. the indicator is inexpensive to compute,

4. the indicator sequence $\left\{I\left(z^k, \Delta z^k\right)\right\}$ converges to its limit vaster than z^k converges to z^*,

5. the indicator gives reliable information early on in the iterative process.

An effective indicator need not satisfy all the properties listed above; however, the sharp separation property is of extreme importance. This is because the indicator function should be able to distinguish between zero variables and those variables with very small values. Due to this, using variables as indicators (by setting variables with very small absolute value to zero) will not work well in our application.

3.2 Tapia Indicator

Tapia [4] suggested using the quotient of successive Lagrange multipliers (λ, μ) and the quotient of successive slack variables (x, s) for indicator functions. The lower bound

indicators corresponding to the primal and dual variables, $I_p^{\mathcal{L}^*}$ and $I_d^{\mathcal{L}^*}$, respectively, are defined as

$$I_p^{\mathcal{L}^*}\left(z^k, \Delta z^k\right) = \frac{x^{k+1}}{x^k}$$
$$I_d^{\mathcal{L}^*}\left(z^k, \Delta z^k\right) = 1 - \frac{\lambda^{k+1}}{\lambda^k}.$$

We similarly define the upper bound indicators as

$$I_p^{\mathcal{U}^*}\left(z^k, \Delta z^k\right) = \frac{s^{k+1}}{s^k}$$
$$I_d^{\mathcal{U}^*}\left(z^k, \Delta z^k\right) = 1 - \frac{\mu^{k+1}}{\mu^k}.$$

Assuming that $\left\{z^k\right\}$ converges to a strictly feasible solution z^*, then it can be shown that for $i = 1, \ldots, n$

$$\lim_{k\to\infty} \frac{x_i^{k+1}}{x_i^k} \longrightarrow \begin{cases} 0, & \text{if } i \in \mathcal{L}^* \\ 1, & \text{if } i \notin \mathcal{L}^* \end{cases} \tag{3.1}$$

$$\lim_{k\to\infty} \left(1 - \frac{\lambda_i^{k+1}}{\lambda_i^k}\right) \longrightarrow \begin{cases} 0, & \text{if } i \in \mathcal{L}^* \\ 1, & \text{if } i \notin \mathcal{L}^* \end{cases} \tag{3.2}$$

$$\lim_{k\to\infty} \frac{s_i^{k+1}}{s_i^k} \longrightarrow \begin{cases} 0, & \text{if } i \in \mathcal{U}^* \\ 1, & \text{if } i \notin \mathcal{U}^* \end{cases} \tag{3.3}$$

$$\lim_{k\to\infty} \left(1 - \frac{\mu_i^{k+1}}{\mu_i^k}\right) \longrightarrow \begin{cases} 0, & \text{if } i \in \mathcal{U}^* \\ 1, & \text{if } i \notin \mathcal{U}^* \end{cases} \tag{3.4}$$

The benefit of the Tapia indicators is that they satisfy all the ideal properties as long as the interior-point method we use will allow the sequence $\left\{I\left(z^k, \Delta z^k\right)\right\}$ to converge superlinearly.

4 Numerical Tests

4.1 Applications

We consider three specific problems from Bretthauer and Shetty's pegging paper [2]. Each of these problems has a closed-form solution to the subproblem (P^k) from section 2.2. Choosing these problems allows us to compare our times to those of Bretthauer and Shetty. A larger list of problems can be found in the survey paper from Patrikkson [7]. Our three applications include: (1) stratified sampling, (2) manufacturing capacity planning, and (3) a quadratic knapsack problem. As is the case in most real world problems, the decision variable is integral.

4.1.1 Stratified Sampling

Stratified sampling requires estimating a population mean μ with a value $\overline{y}$ by minimizing the variance of the estimate subject to a linear sampling budget constraint. A population is split into n strata and ξ_i is the sample size for stratum i. If b_i is the cost of surveying one unit in stratum i and M is the sampling budget available, then we may represent this as a nonlinear resource allocation problem:

$$\begin{aligned} \text{(SAMP)} \qquad \min \quad & V(\overline{y}) \\ \text{st} \quad & \sum b_i \xi_i \leqslant M \\ & \ell_i \leqslant \xi_i \leqslant u_i \\ & \xi_i \text{ is integral.} \end{aligned}$$

In regards to pegging, (SAMP) lies under Case 2.

4.1.2 Manufacturing Capacity Planning

Many manufacturing systems can be characterized by the machines or workstations in the network. Typically this is a network of queues where each node in the network is a workstation. The manufacturing capacity planning problem involves the minimum cost selection of the capacity at each workstation subject to an upper limit on the total dollar value of work-in-process in the system [2]. It can be formulated as follows:

$$\begin{aligned} \text{(MCP)} \qquad \min \quad & \sum c_i \xi_i \\ \text{st} \quad & \sum b_i \left(\frac{\alpha_i}{\xi_i - \alpha_i} \right) \leqslant M \\ & \ell_i \leqslant \xi_i \leqslant u_i \\ & \xi_i \text{ is integral} \end{aligned}$$

where n is the number of stations in the network, ξ_i the service rate at station i, c_i the cost per unit of capacity at station i, b_i the average dollar work-in-process value per job at station i, α_i the arrival rate at station i, and M an upper limit on total work-in-process allowed in the network. In regards to pegging, (MCP) lies under Case 1.

4.1.3 Quadratic Knapsack

A standard optimization problem which requires minimizing a quadratic objective function subject to a linear capacity constraint.

$$\begin{aligned} \text{(QP)} \qquad \min \quad & \sum \left(\frac{1}{2} p_i \xi_i^2 - a_i \xi_i \right) \\ \text{st} \quad & \sum b_i \xi_i \leqslant M \\ & \ell_i \leqslant \xi_i \leqslant u_i \\ & \xi_i \text{ is integral.} \end{aligned}$$

In regards to pegging, (QP) lies under Case 2.

4.2 Implementation

We implemented several different algorithms using the methods discussed earlier. All of these methods were coded in MATLAB.

PEG: A pegging algorithm adapted from Bretthauer and Shetty [2] with optimizations from the paper by Kiwiel [5].

IPM-PC: A linearly convergent predictor-correct interior-point method with an optimized Newton system solver as discussed in section 2.3.2.

IPMQ: A quadratically convergent interior-point method from Sun and Zhao [9] with an optimized Newton system solver.

IPMQ-PEG: A synthesis of the **IPMQ** and **PEG** methods using Tapia indicators from section 3.2.

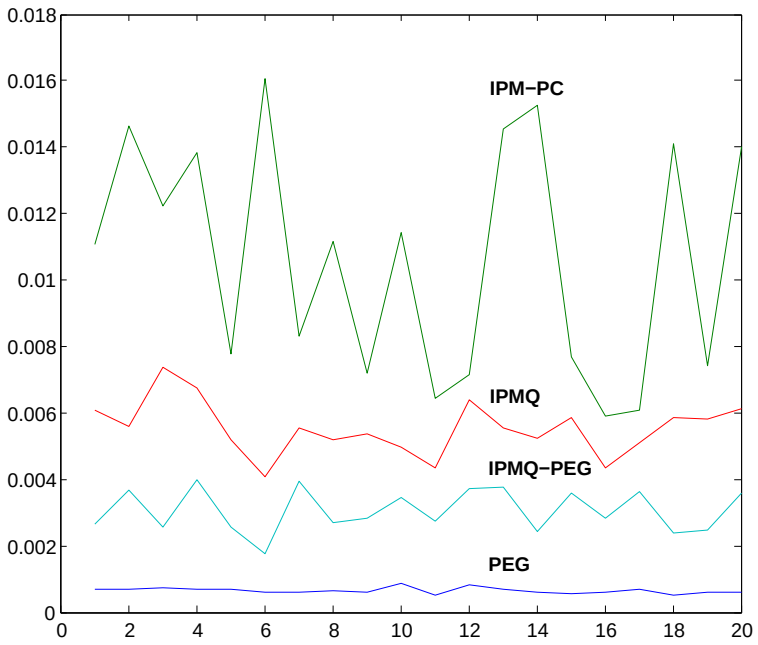
The method of interest to us is **IPMQ-PEG**. Since problem (P) can be solved using a pegging method **PEG**, it was only natural to try to identify which variables are pegged early on in an iterative process like an interior-point method. To identify these variables, we attempted to use Tapia indicators.

The **IPMQ-PEG** method checks the Tapia indicators every iteration and then decides whether the iterates $\left\{ I\left(z^k, \Delta z^k\right) \right\}$ have converged. Once they converged, the method populates the index sets L^1 and U^1 (from Algorithm 1) with the respective variables pegged to the lower and upper bounds, then runs **PEG** to find a solution.

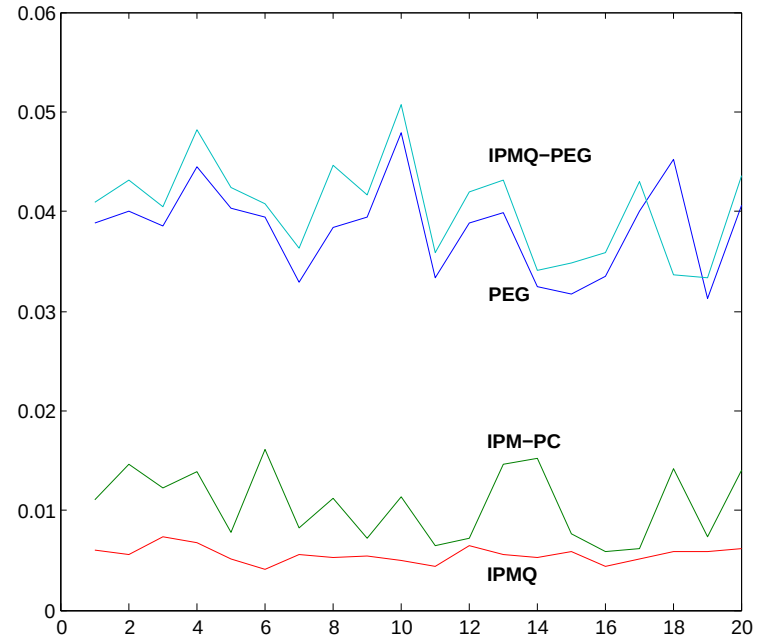
4.3 Timings

Our numerical tests showed that **PEG** is among the fastest methods to solve (P) as long as a closed-form solution exists to the subproblem (P^k) which can be seen in Figure 3(a). The

new implementation **IPMQ-PEG** is competitive with the existing interior-point methods and its timings correlate strongly with **PEG**.



(a) A closed-form solution to the pegging subproblem exists and is used in **PEG**.



(b) A numerical solver used to solve the pegging subproblem slows down any method using **PEG**.

Figure 3: *Timing plots for both a closed-form and numerical solution to the pegging subproblem. The horizontal axis represents one of the twenty timings and the vertical axis is the timing in seconds.*

If such a numerical solution does not exist, then we see in Figure 3(b) that **PEG** and **IPMQ-PEG** are slower than the other two interior-point methods. Both **PEG** and **IPMQ-PEG** can be significantly sped up using more efficient numerical subsolvers to solve (P^k).

The timings in Figure 3(a) and Figure 3(b) are for small problems with $n = 40$. We found the timings were extremely similar for any of the three problem applications we discussed in section 4.1. For larger dimensional problems (like $n = 5000$) we find that **IPMQ**, **PEG**, and **IPMQ-PEG** require the same number of iterations as smaller dimensional problems but the amount of work scales linearly with n.

4.4 Future Work

Among the methods studied, we only briefly mentioned breakpoint search methods in section 2.1. Any future timings should be compared to an efficient breakpoint search method such as one done by Kiwiel [5]. To speed up the algorithms already implemented, there are still some small optimizations from Kiwiel's paper that have yet to be placed in the existing code.

So far we have only studied three major nonlinear resource allocation problems where each of these problems had a closed-form solution to the pegging subproblem (P^k). The existence of such a solution shows that **PEG** dominated all of the other methods implemented. In the future we should consider problems that only have numerical solutions to the subproblem. Furthermore, the three problems studied had almost half the variables pegged within the first two iterations of **PEG**. It should be possible to find applications of problem (P) in which this does not happen or find a specific example where convergence of the pegging algorithm is slower than **IPMQ-PEG**.

The paper by El-Bakry, Tapia, and Zhang [4] concerned itself with indicators for interior-point methods. Although numerous indicator functions were mentioned, we only considered the Tapia indicators because they were easy to implement. We should try using other indicators such as the Tapia-Zhang indicators to see if they identify active constraints faster than in our current implementation.

References

[1] Bitran GR, Hax AC. *Disaggregation and Resource Allocation Using Convex Knapsack Problems with Bounded Variables*. Management Science, Vol. 27. (1981), pp. 431-41.

[2] Bretthauer K, Shetty B. *The Nonlinear Resource Allocation Problem*. Operations Research, Vol. 43, No. 4. (1995), pp. 670-83.

[3] El-Bakry AS. *On the Role of Indicators in Identifying Zero Variables in Linear Programming*. Doctoral Thesis, Rice University. (1991).

[4] El-Bakry AS, Tapia RA, Zhang Y. *A study of indicators for identifying zero variables in interior-point methods*. SIAM Review, Vol. 36 No. 1. (March 1994), pp. 45-72.

[5] Kiwiel K. *Variable Fixing Algorithms for the Continuous Quadratic Knapsack Problem*. Journal of Optimization Theory and Applications, Vol. 136, No. 3. (2008), pp. 445-58.

[6] Koopman, BO. *The optimum distribution of effort*. Operations Research, No. 1. (1953), pp. 52-63.

[7] Patriksson M. *A survey on the continuous nonlinear resource allocation problem*. European Journal Operational Research, Vol. 185, No. 1. (2008), pp. 1-46.

[8] Robinson AG, Jiang N, Lerme CS. *On the continuous quadratic knapsack problem*. Mathematical Programming, Vol. 55. (1992), pp. 99-108.

[9] Sun J, Zhao G. *A quadratically convergent polynomial long-step algorithm for a class of nonlinear monotone complementarity problems*. Optimization, Vol. 48, No. 4. (2000), pp. 453-75.

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