Hindawi
Journal of Control Science and Engineering
Volume 2024, Article ID 7602027, 17 pages
https://doi.org/10.1155/2024/7602027

Research Article

An Accelerated Fixed-Point Algorithm Applied to Quadratic Convex Separable Knapsack Problems

Atécio Alves,[1] **Jônatas O. L. Silva**,[2] **Luiz C. Matioli**,[3] **Paulo S. M. Santos**,[4] **and Sissy S. Souza**[4]

[1]*Programa de Pós-Graduação em Matemática, Federal University of Piauí, Teresina, Brazil*
[2]*Programa de Pós-Graduação Doutorado em Ciência da Computação-Associação UFMA/UFPI, Federal University of Maranhão, São Luís, Maranhão, Brazil*
[3]*Departamento de Matemática, Federal University of Paraná, Curitiba, Brazil*
[4]*Coordenação do Curso de Matemática, Federal University of Delta do Parnaíba, Parnaíba, Brazil*

Correspondence should be addressed to Jônatas O. L. Silva; jonatas.iw@gmail.com

Received 3 May 2023; Revised 7 November 2023; Accepted 27 November 2023; Published 8 February 2024

Academic Editor: Radek Matušů

In this article, we propose a root-finding algorithm for solving a quadratic convex separable knapsack problem, which is more straightforward than existing methods and competitive in practice. Besides, we also present an extension of the proposal, which improves its computational time, and then we incorporate the accelerated Anderson's and Aitken's fixed-point algorithms to obtain better results. The algorithm only performs function evaluations. We present partial convergence results of the algorithm. Moreover, we illustrate superior computational results in medium and large problems as well as the applicability of the algorithm with real-life applications to show their efficiency.

1. Introduction

We are interested in solving the quadratic convex separable knapsack problem:

$$(\text{QSKP}): \begin{cases} \text{Minimize } f(x) := \sum_{i=1}^{n}\left(\frac{1}{2}p_i x_i^2 - a_i x_i\right), \\ s.t. \sum_{i=1}^{n} b_i x_i = c, \\ l_i \le x_i \le u_i, i = 1, 2, \ldots, n, \end{cases} \tag{1}$$

where $p_i, b_i > 0$ and $l_i < u_i$ for all $i = 1, 2, \ldots, n$ with $c > 0$ such that $\langle b, l\rangle \le c \le \langle b, u\rangle$.

Problem (1) has a variety of applications, including among them, financial models, production and inventory management, stratified sampling, optimal design of queuing network models in manufacturing, computer systems, subgradient optimization, and health care (see[1–6] and references therein).

Motivated by its extensive applications, a significant amount of attention has been attracted to developing optimization algorithms, and many iterative methods have been proposed to solve this simple problem. See, for instance, an excellent survey [4], which is complemented by [5]. In addition, they contribute to an improvement in the process of fixing variables in the relaxation algorithm and a better way to evaluate subsolutions. Finally, they provided a rigorous numerical evaluation of several relaxations (primal) and breakpoint (dual) algorithms, incorporating a variety of pegging strategies as well as a Newton-type method.

This article aims to propose a fixed-point algorithm (FPA) capable of solving the box-constrained quadratic convex separable knapsack (QSKP) problem as efficiently as other state-of-the-art methods. We extend the FPA to apply in the quadratic convex separable knapsack problem under upper bound constraints (QSKPz). After reformulating the QSKP problem, the new problem QSKPz is solved by an extension of the FPA, called FPA2, which requires fewer calculations than applying FPA directly in the QSKP

problem. To obtain a better performance of the proposed methods, we also incorporated the Anderson algorithm [7] and a generalization of the Aitken algorithm [8] for fixed-point methods to improve the proposal. Such acceleration approaches are also explored in [9–11]. Below, we briefly describe each section of this article.

First, in Section 2, we define the proposed method, which is based on [3]. While this article modified the method to solve quadratic convex separable knapsack problems, in [3], the fixed-point method solves problems in stratified sampling under box constraints. In Section 3, we propose the algorithm and establish its convergence. In Section 4, we extend the proposed fixed-point algorithm to be applied to quadratic convex separable knapsack problem under upper bound constraints. We describe the strategy to accelerate the fixed-point convergence in Section 5 and in Section 6 we present the numerical experiments. We finally present the final remarks in Section 7.

2. Fixed-Point Method

This section provides the material necessary for article comprehension. We start by proposing the existence and uniqueness of the solution to problem (1). Here, we rephrased x as a function dependent on λ. This reformulation leads to a root-finding problem for the Lagrange multiplier of the equality constraint, which is needed to formulate the fixed-point iteration as in [3].

Firstly, based on [1], we present an optimality condition theorem.

Theorem 1. *A vector $x^* \in \mathbb{R}^n$ is a minimum of problem (1) if and only if there exist Lagrange multipliers $\lambda_* \in \mathbb{R}$, $v^* \in \mathbb{R}^n_+$, and $w^* \in \mathbb{R}^n_+$ such that*

$$\nabla L_x(x^*, \lambda_*, v^*, w^*) = \nabla f(x^*) + \lambda_* \nabla g(x^*) + (\nabla r(x^*))' v^* + (\nabla s(x^*))' w^* = 0, \tag{2}$$

and furthermore,

$$v_i r_i(x_i^*) = 0, \quad i = 1, \cdots, n, \tag{3}$$

$$w_i s_i(x_i^*) = 0, \quad i = 1, \cdots, n, \tag{4}$$

where $g(x) = b^T x - c$, $r(x) = l - x$, and $s(x) = x - u$.

Lemma 2. *Under the given assumptions of Theorem 1, equation (2) is equivalent to*

$$\left.\begin{aligned} 0 &\geq \frac{p_i u_i - a_i}{b_i} + \lambda_*, && \text{if } x_i^* = u_i, \\ 0 &= \frac{p_i x_i^* - a_i}{b_i} + \lambda_*, && \text{if } x_i^* \in (l_i, u_i), \\ 0 &\leq \frac{p_i l_i - a_i}{b_i} + \lambda_*, && \text{if } x_i^* = l_i \end{aligned}\right\}, \tag{5}$$

for all $i = 1, 2, \ldots, n$.

Proof. Let $x^* \in \mathbb{R}^n$ be a solution of problem (1). If $x_i^* = u_i$, by (3), $v_i^* = 0$ and $w_i^* \geq 0$. By (2) and for all $i = 1, 2, \ldots, n$, we have

$$\begin{aligned} 0 &= p_i x_i^* - a_i + b_i \lambda_* - v_i^* + w_i^* \\ &= p_i u_i - a_i + b_i \lambda_* + w_i^* \geq p_i u_i - a_i + b_i \lambda_*. \end{aligned} \tag{6}$$

Then,

$$0 \geq \frac{p_i u_i - a_i}{b_i} + \lambda_*. \tag{7}$$

If $x_i^* \in (l_i, u_i)$, by (3) and (4), $v_i^* = 0$ and $w_i^* = 0$. By (2) and for all $i = 1, 2, \ldots, n$, we have

$$0 = p_i x_i^* - a_i + b_i \lambda_* - v_i^* + w_i^* = p_i x_i^* - a_i + b_i \lambda_*. \tag{8}$$

Then,

$$0 = \frac{p_i x_i^* - a_i}{b_i} + \lambda_*. \tag{9}$$

If $x_i^* = l_i$, by (4), $w_i^* = 0$ and $v_i^* \geq 0$. By (2) and for all $i = 1, 2, \ldots, n$, we have

$$\begin{aligned} 0 &= p_i x_i^* - a_i + b_i \lambda_* - v_i^* + w_i^* \\ &= p_i l_i - a_i + b_i \lambda_* - v_i^* \leq p_i l_i - a_i + b_i \lambda_*. \end{aligned} \tag{10}$$

Then,

$$0 \leq \frac{p_i l_i - a_i}{b_i} + \lambda_*. \tag{11}$$

Thus, we conclude the proof.

Motivated by (5), we will treat x as a function depending on the variable λ. To achieve this, we set

$$x\colon \mathbb{R} \longrightarrow \mathbb{R}^n, \lambda \longmapsto x(\lambda), \tag{12}$$

with

$$x_i(\lambda) = \begin{cases} l_i, & \text{if } \lambda \geq \dfrac{a_i - p_i l_i}{b_i}, \\ \dfrac{a_i - \lambda b_i}{p_i}, & \text{if } \dfrac{a_i - p_i u_i}{b_i} < \lambda < \dfrac{a_i - p_i l_i}{b_i}, \\ u_i, & \text{if } \lambda \leq \dfrac{a_i - p_i u_i}{b_i}. \end{cases} \tag{13}$$

□

Theorem 3. *A vector $x^* \in \mathbb{R}^n$ is the unique solution of optimization problem (1) if and only if there exists a multiplier $\lambda_* \in \mathbb{R}$ such that$x(\lambda_*)$ defined in (13) satisfies*

$$g(x(\lambda_*)) = 0. \tag{14}$$

Proof. Let $x^* \in \mathbb{R}^n$ be a solution of problem (1); then, $g(x^*) = 0$. Considering (x^*, λ_*) so that it satisfies (5), we define $x(\lambda_*)$ as in (6). Thus, if $x_i^* = l_i$, we have

$$0 \le \frac{p_i l_i - a_i}{b_i} + \lambda_* \iff \lambda_* \ge \frac{a_i - p_i l_i}{b_i}, \tag{15}$$

and then $x_i(\lambda_*) = l_i$, i.e., $x_i^* = x_i(\lambda_*)$. If $x_i^* \in (l_i, u_i)$, we have

$$\begin{aligned} 0 &= \frac{p_i x_i^* - a_i}{b_i} + \lambda_* \iff x_i^* = \frac{a_i - b_i\lambda_*}{p_i}, \\ l_i &< \frac{a_i - b_i\lambda_*}{p_i} < u_i \iff \frac{a_i - p_i u_i}{b_i} < \lambda_* < \frac{a_i - p_i l_i}{b_i}, \end{aligned} \tag{16}$$

and then $x_i(\lambda_*) = a_i - b_i\lambda_*/p_i$, i.e., $x_i^* = x_i(\lambda_*)$. If $x_i^* = u_i$, we have

$$0 \ge \frac{p_i u_i - a_i}{b_i} + \lambda_* \iff \lambda_* \le \frac{a_i - p_i u_i}{b_i}, \tag{17}$$

and then $x_i(\lambda_*) = u_i$, i.e., $x_i^* = x_i(\lambda_*)$. Thus, we conclude that $x^* = x(\lambda_*)$. On the other hand, for some λ_* the vector $x(\lambda_*)$ satisfies

$$g(x(\lambda_*)) = 0. \tag{18}$$

It is easy to check that $(x(\lambda_*), \lambda_*)$ also satisfies (5). The proof is complete.

Now, to get a fixed-point based algorithm formulation, we use (13) to define

$$\begin{aligned} I_\ell(\lambda) &:= \left\{ i \in \{1, 2, \ldots, n\} \colon \lambda \ge \frac{a_i - p_i l_i}{b_i} \right\}, \\ I_u(\lambda) &:= \left\{ i \in \{1, 2, \ldots, n\} \colon \lambda \le \frac{a_i - p_i u_i}{b_i} \right\}, \\ I_{\mathrm{eq}}(\lambda) &:= \{1, 2, \ldots, n\} \setminus (I_\ell(\lambda) \cup I_u(\lambda)). \end{aligned} \tag{19}$$

We have that

$$\begin{aligned} g(x(\lambda)) &= b^T x(\lambda) - c \\ &= \sum_{i \in I_\ell(\lambda)} b_i l_i + \sum_{i \in I_u(\lambda)} b_i u_i + \sum_{i \in I_{\mathrm{eq}}(\lambda)} b_i \frac{a_i - \lambda b_i}{p_i} - c \\ &= \sum_{i \in I_\ell(\lambda)} b_i l_i + \sum_{i \in I_u(\lambda)} b_i u_i + \sum_{i \in I_{\mathrm{eq}}(\lambda)} \left(\frac{b_i a_i}{p_i} - \frac{\lambda b_i^2}{p_i} \right) - c \\ &= \sum_{i \in I_\ell(\lambda)} b_i l_i + \sum_{i \in I_u(\lambda)} b_i u_i + \sum_{i \in I_{\mathrm{eq}}(\lambda)} \frac{b_i a_i}{p_i} - \lambda \sum_{i \in I_{\mathrm{eq}}(\lambda)} \frac{b_i^2}{p_i} - c. \end{aligned} \tag{20}$$

Then, $g(x(\lambda)) = 0$ if and only if

$$\lambda \sum_{i \in I_{\mathrm{eq}}(\lambda)} \frac{b_i^2}{p_i} = \sum_{i \in I_\ell(\lambda)} b_i l_i + \sum_{i \in I_u(\lambda)} b_i u_i + \sum_{i \in I_{\mathrm{eq}}(\lambda)} \frac{b_i a_i}{p_i} - c, \tag{21}$$

that is,

$$\lambda = \frac{\sum_{i \in I_\ell(\lambda)} b_i l_i + \sum_{i \in I_u(\lambda)} b_i u_i + \sum_{i \in I_{\mathrm{eq}}(\lambda)} (b_i a_i / p_i) - c}{\sum_{i \in I_{\mathrm{eq}}(\lambda)} b_i^2 / p_i}. \tag{22}$$

Then, we define the following function:

$$\begin{aligned} &F \colon \mathbb{R} \longrightarrow \mathbb{R}, \\ &\lambda \longmapsto F(\lambda) = \frac{\sum_{i \in I_\ell(\lambda)} b_i l_i + \sum_{i \in I_u(\lambda)} b_i u_i + \sum_{i \in I_{\mathrm{eq}}(\lambda)} (b_i a_i / p_i) - c}{\sum_{i \in I_{\mathrm{eq}}(\lambda)} b_i^2 / p_i}. \end{aligned} \tag{23}$$

It is easy to see that $g(x(\lambda)) = 0$ if, and only if, $\lambda = F(\lambda)$. □

Remark 4. Formula (22) appears in [12] as an intermediate step in the variable-fixing algorithms. Moreover, following [12], *we can assume that for all* $k \in \mathbb{N}$, if $I_{\mathrm{eq}}(\lambda_k) \neq \varnothing$, then $I_{\mathrm{eq}}(\lambda_{k+1}) \neq \varnothing$.

3. Statement of Fixed-Point Algorithm and Its Convergence Results

Now, we formally describe the fixed-point-based Algorithm 1 (abbreviated as FPA). For sake of simplicity, for $k \in \mathbb{N}$, we denote $I_\ell(\lambda^k) = I_\ell^k$, $I_u(\lambda^k) = I_u^k$, and $I_{\mathrm{eq}}(\lambda^k) = I_{\mathrm{eq}}^k$.

Kim and Wu [6] proposed an improvement characterized by eliminating calculations of all primal variables in every iteration as in [2, 12]. The natural formulation of the FPA algorithm (i.e., equation (13)) leads to the improvement proposed in [6]. Besides, the FPA algorithm does not necessarily need a variable fixing step as in the other state-of-the-art methods, although we can implement it. We aim to keep the FPA algorithm as simple as possible and apply an extension capable of improving its performance, as the acceleration step presented in the next section.

Below, the first result concerns the algorithm's stop criteria.

Proposition 5. *If the FPA generates a finite sequence, then the last point is a solution of problem (1).*

Proof. Let us assume that $\lambda_{k+1} = \lambda_k \in \mathbb{R}$ is the last point obtained by the proposed algorithm. So, we have

$$\begin{aligned} \lambda_k &= \frac{\sum_{i \in I_\ell^k} b_i l_i + \sum_{i \in I_u^k} b_i u_i + \sum_{i \in I_{\mathrm{eq}}^k} (b_i a_i / p_i) - c}{\sum_{i \in I_{\mathrm{eq}}^k} b_i^2 / p_i}, \\ \sum_{i \in I_{\mathrm{eq}}^k} \frac{b_i^2}{p_i} \lambda_k &= \sum_{i \in I_\ell^k} b_i l_i + \sum_{i \in I_u^k} b_i u_i + \sum_{i \in I_{\mathrm{eq}}^k} \frac{b_i a_i}{p_i} - c. \end{aligned} \tag{24}$$

Step 0 (Initialization)
Set k = 0. Let $\lambda^k \in \mathbb{R}$ according to [2, 6, 12].
Step 1 (Calculating dual bounds)
For $i \in \{1, 2, \ldots, n\}$,
Compute $[\mathrm{LR}_i, \mathrm{UR}_i] = [a_i - p_i l_i/b_i, a_i - p_i u_i/b_i]$.
Step 2 (Calculating fixed-point sums)
$\mathrm{SLR}^k = \sum_{i \in I_l^k}^n b_i l_i$, where $I_l^k = \{i\colon \lambda^k \geq \mathrm{LR}_i, i \in \{1, 2, \ldots, n\}\}$.
$\mathrm{SLU}^k = \sum_{i \in I_u^k}^n b_i u_i$, where $I_u^k = \{i\colon \lambda^k \leq \mathrm{UR}_i, i \in \{1, 2, \ldots, n\}\}$.
$\mathrm{SLE}_a^k = \sum_{i \in I_{\mathrm{eq}}^k}^n b_i a_i/p_i$, where $I_{\mathrm{eq}}^k = \{i\colon \mathrm{UR}_i < \lambda^k < \mathrm{LR}_i, i \in \{1, 2, \ldots, n\}\}$.
$\mathrm{SLE}_b^k = \sum_{i \in I_{\mathrm{eq}}^k}^n b_i^2/p_i$, where $I_{\mathrm{eq}}^k = \{i\colon \mathrm{UR}_i < \lambda^k < \mathrm{LR}_i, i \in \{1, 2, \ldots, n\}\}$.
Step 3 (Update dual variable)
Compute $\lambda^{k+1} = \mathrm{SLR}^k + \mathrm{SLU}^k + \mathrm{SLE}_a^k - c/\mathrm{SLE}_b^k$.
Step 4 (Check stopping criterion)
If $\mathrm{abs}(\lambda^{k+1} - \lambda^k) < \epsilon$, then set x_i according to equation (13) and STOP. Otherwise, set $k = k + 1$ and return to **Step 2**.

Algorithm 1: FPA algorithm.

It implies that

$$0 = \sum_{i \in I_\ell^k} b_i l_i + \sum_{i \in I_u^k} b_i u_i + \sum_{i \in I_{\mathrm{eq}}^k} \frac{b_i a_i}{p_i} - \sum_{i \in I_{\mathrm{eq}}^k} \frac{b_i^2}{p_i} \lambda_k - c = g(x(\lambda_k)). \tag{25}$$

Hence, our conclusion follows from Theorem 3.

From now on, we assume that the FPA algorithm generates an infinite sequence denoted by $\{\lambda_k\}$, and we present the following important properties. □

Proposition 6. *The sequence $\{\lambda_k\}$ is bounded, that is, there exists $M > 0$ such that $|\lambda_k| \leq M$ for all $k \in \mathbb{N}$.*

Proof. In fact, for all $k \in \mathbb{N}$, we have

$$\begin{aligned} |\lambda_k| &= \left| \frac{\sum_{i \in I_\ell^{k-1}} b_i l_i + \sum_{i \in I_u^{k-1}} b_i u_i + \sum_{i \in I_{\mathrm{eq}}^{k-1}} (b_i a_i/p_i) - c}{\sum_{i \in I_{\mathrm{eq}}^{k-1}} b_i^2/p_i} \right| \\ &\leq \left| \frac{\sum_{i \in I_\ell^{k-1}} b_i l_i + \sum_{i \in I_u^{k-1}} b_i u_i + \sum_{i \in I_{\mathrm{eq}}^{k-1}} (b_i a_i/p_i) - c}{\min\{b_i^2/p_i\colon 1 \leq i \leq n\}} \right| \\ &\leq \frac{\|b\| (\|\ell\| + \|u\|) + |n.\max\{(b_i a_i/p_i)\colon 1 \leq i \leq n\}| + |c|}{\min\{b_i^2/p_i\colon 1 \leq i \leq n\}} =: M. \end{aligned} \tag{26}$$

As in [3], in the following result, we assume that the inequalities used in the definition of $I_\ell(\lambda)$ and $I_u(\lambda)$ are strict. Then, we show that small perturbations of λ_* do not change these index sets. It means that if the fixed-point iteration converges, then the iteration terminates after finitely many steps with the exact solution λ_* because the index sets do not change anymore. □

Lemma 7. *Consider λ_* such that $\lambda_* \notin \{a_i - p_i l_i/b_i, a_i - p_i u_i/b_i\colon i = 1, \ldots, n\}$. Then, there exists $\epsilon > 0$ such that if $|\lambda - \lambda_*| < \epsilon$, we have*

$$\begin{aligned} I_\ell(\lambda) &= I_\ell(\lambda_*), \\ I_u(\lambda) &= I_u(\lambda_*), \\ I_{\mathrm{eq}}(\lambda) &= I_{\mathrm{eq}}(\lambda_*). \end{aligned} \tag{27}$$

Proof. From our assumptions, we may rewrite

$$\begin{aligned} I_\ell(\lambda_*) &= \left\{ i \in \{1, 2, \ldots, n\}\colon \lambda_* > \frac{a_i - p_i l_i}{b_i} \right\}, \\ I_u(\lambda_*) &= \left\{ i \in \{1, 2, \ldots, n\}\colon \lambda_* < \frac{a_i - p_i u_i}{b_i} \right\}. \end{aligned} \tag{28}$$

Hence, we get our aim by considering

$$\epsilon := \min\left\{ \left|\lambda_* - \frac{a_i - p_i u_i}{b_i}\right|, \left|\lambda_* - \frac{a_i - p_i l_i}{b_i}\right|\colon i = 1, \ldots, n \right\}. \tag{29}$$

□

4. FPA Extended to Quadratic Convex Separable Knapsack under Upper Bound Constraints

This section presents a variable substitution in problem (1) to obtain a box constraint of the type $0 \leq z_i \leq u_i'$, $i = 1, 2, \ldots, n$. Besides speeding up the computational time of problem (1), such formulation has direct applications in the continuous relaxation of the sensor placement problem [13] and problems arising in multicommodity network flow and logistics, as presented in [14, 15].

Defining

$$x_i = z_i + l_i, \tag{30}$$

and substituting in (1), we have

$$\begin{aligned}\frac{1}{2}p_i x_i^2 - a_i x_i &= \frac{1}{2}p_i (z_i + l_i)^2 - a_i (z_i + l_i)\\ &= \frac{1}{2}p_i z_i^2 - (a_i - p_i l_i) z_i + \left(\frac{1}{2}p_i l_i^2 - a_i l_i\right),\\ \sum_{i=1}^{n} x_i b_i = c &\Longleftrightarrow \sum_{i=1}^{n} (z_i + l_i) b_i = c \Longleftrightarrow \sum_{i=1}^{n} z_i b_i = c - \sum_{i=1}^{n} b_i l_i,\\ l_i \le x_i \le u_i &\Longleftrightarrow 0 \le x_i - l_i \le u_i - l_i \Longleftrightarrow 0 \le z_i \le u_i - l_i,\end{aligned} \tag{31}$$

and then problem (1) is equivalent to the following problem:

$$\text{(QSKPz)}: \begin{cases} \text{Minimize} f_1(x) := \displaystyle\sum_{i=1}^{n}\left(\frac{1}{2}p_i z_i^2 - a_i' z_i\right), \\ s.t. \displaystyle\sum_{i=1}^{n} b_i z_i = c', \\ l_i' \le z_i \le u_i', i = 1, 2, \ldots, n, \end{cases} \tag{32}$$

where $z_i = x_i - l_i$, $a_i' = a_i - p_i l_i$, $c' = c - \sum_{i=1}^{n} b_i l_i$, $u_i' = u_i - l_i$, and l' is a vector of zeros.

Now, through (13) we can write a function z_i depending on the variable λ as follows:

$$z_i(\lambda) = \begin{cases} l', & \text{if } \lambda \ge \dfrac{a_i'}{b_i}, \\ \dfrac{a_i' - \lambda b_i}{p_i} - l_i, & \text{if } \dfrac{a_i' - p_i u_i'}{b_i} < \lambda < \dfrac{a_i'}{b_i}, \\ u', & \text{if } \lambda \le \dfrac{a_i' - p_i u_i'}{b_i}. \end{cases} \tag{33}$$

To get the new fixed-point algorithm, we may have to define the sets $I_l'(\lambda)$, $I_u'(\lambda)$, and $I_{\text{eq}}'(\lambda)$ according to (19).

$$\begin{aligned} I_\ell'(\lambda) &:= \left\{ i \in \{1, 2, \ldots, n\} : \lambda \ge \frac{a_i'}{b_i} \right\}, \\ I_u'(\lambda) &:= \left\{ i \in \{1, 2, \ldots, n\} : \lambda \le \frac{a_i' - p_i u_i'}{b_i} \right\}, \\ I_{\text{eq}}'(\lambda) &:= \{1, 2, \ldots, n\} \setminus \left(I_\ell'(\lambda) \cup I_u'(\lambda)\right). \end{aligned} \tag{34}$$

Then, by equations (20) and (22), since the lower bound of the reformulated problem (32) is a vector of zeros, we can assume that $\sum_{i \in I_\ell'} b_i l_i' = 0$. Therefore, we can define the new λ as

$$\lambda = \frac{\sum_{i \in I_u'(\lambda)} b_i u_i' + \sum_{i \in I_{\text{eq}}'(\lambda)} (b_i a_i' / p_i) - c}{\sum_{i \in I_{\text{eq}}'(\lambda)} b_i^2 / p_i}. \tag{35}$$

Defining the new function

$$F' : \mathbb{R} \longrightarrow \mathbb{R}, \tag{36}$$

we obtain

$$\lambda \longmapsto F'(\lambda) = \frac{\sum_{i \in I_u'(\lambda)} b_i u_i' + \sum_{i \in I_{\text{eq}}'(\lambda)} (b_i a_i' / p_i) - c}{\sum_{i \in I_{\text{eq}}'(\lambda)} b_i^2 / p_i}. \tag{37}$$

According to the formulations above, we describe the new fixed-point-based Algorithm 2 (FPA2) below.

The main advantage of the FPA2 algorithm proposed above is in Step 2, where a sum presented in the FPA algorithm is no longer needed in this new algorithm. This modification makes the algorithm even simpler and makes it possible to improve its performance, as we will see in the experiments section. The convergence analysis of the FPA2 algorithm follows as shown in Section 3.

5. Fixed-Point Acceleration

As mentioned in [11], acceleration methods can alleviate slow convergence. Our interest here is in two particular acceleration methods. The first originated from the work of [7], which we refer to as Anderson acceleration, and the second one resulted from the work of [8], which we refer to as Aitken acceleration. In the following subsections, we describe both algorithms and define the accelerated fixed-point method incorporating the acceleration techniques.

5.1. Anderson Acceleration. Anderson's acceleration defines a vector of weights $\alpha \in \mathbb{R}^m$. These weights are determined using the following optimization problem:

$$\begin{aligned} &\min_\alpha \|G_k \alpha\|_2, \\ &s.t. \sum_{i=0}^{m} \alpha_i = 1, \end{aligned} \tag{38}$$

where G is found as follows. Let us consider $\lambda = F(\lambda)$ according to equation (23) and $g = F(\lambda) - \lambda$. Then, $G_k = [g_{k-m}, \ldots, g_i]$, where $g_k = F(\lambda_k) - \lambda_k$.

With these weights, we are able to create the expression of the next iteration as

$$\lambda_{k+1} = \sum_{i=0}^{m} \alpha_i F(\lambda_{k-m+i}). \tag{39}$$

To improve the fixed-point algorithm's convergence rate, we consider the Anderson acceleration algorithm formulated as in [11]. Below, we describe the Anderson approach incorporated into FPA2 Algorithm 3.

We see that we must solve a constrained minimization problem at each iteration. In most references, the minimization in equation (38) is recast as an unconstrained minimization problem. We generally keep the number of elements, m, in the Anderson history small to ensure we have sufficient storage and make the optimization problem less ill-conditioned. In our experiments, we define $m = 2$. More about Anderson's acceleration theory and formulations can be seen in [9–11].

Step 0 (Initialization)
Set $k=0$. Let $\lambda^k \in \mathbb{R}$ according to [2, 6, 12].
Step 1 (Calculating dual bounds)
For $i \in \{1, 2, \ldots, n\}$,
Compute $[\mathrm{LR}_i, \mathrm{UR}_i] = [a_i'/b_i, a_i' - p_i u_i'/b_i]$.
Step 2 (Calculating fixed-point sums)
$\mathrm{SLU}^k = \sum_{i \in I_u'^k}^n b_i u_i'$, where $I_u'^k = \{i\colon \lambda^k \le \mathrm{UR}_i, i \in \{1, 2, \ldots, n\}\}$.
$\mathrm{SLE}_a^k = \sum_{i \in I_{\mathrm{eq}}'^k}^n b_i a_i'/p_i$, where $I_{\mathrm{eq}}'^k = \{i\colon \mathrm{UR}_i < \lambda^k < \mathrm{LR}_i, i \in \{1, 2, \ldots, n\}\}$.
$\mathrm{SLE}_b^k = \sum_{i \in I_{\mathrm{eq}}'^k}^n b_i^2/p_i$, where $I_{\mathrm{eq}}'^k = \{i\colon \mathrm{UR}_i < \lambda^k < \mathrm{LR}_i, i \in \{1, 2, \ldots, n\}\}$.
Step 3 (Update dual variable)
Compute $\lambda^{k+1} = \mathrm{SLU}^k + \mathrm{SLE}_a^k - c/\mathrm{SLE}_b^k$.
Step 4 (Check stopping criterion)
If $\mathrm{abs}(\lambda^{k+1} - \lambda^k) < \epsilon$, then set x_i according to equations (30) and (33) and STOP. Otherwise, set $k = k + 1$ and return to **Step 2**.

Algorithm 2: FPA2 algorithm.

Step 0 (Initialization)
Set $k=0$. Let $\lambda^k \in \mathbb{R}$ according to [2, 6, 12].
Define $m > 0$.
Step 1 (Calculating dual bounds)
For $i \in \{1, 2, \ldots, n\}$,
Compute $[\mathrm{LR}_i, \mathrm{UR}_i] = [a_i'/b_i, a_i' - p_i u_i'/b_i]$.
Step 2 (Compute $G(\lambda^k)$)
Compute SLU^k, SLE_a^k and SLE_b^k using λ^k according to FPA2 algorithm.
Compute $G(\lambda^k) = \mathrm{SLU}^k + \mathrm{SLE}_a^k - c/\mathrm{SLE}_b^k$.
Step 3 (Updating Anderson acceleration variables)
Set $m_k = \min\{m, k\}$.
Let $f_k = G(\lambda^k) - \lambda^k$.
Set $F_k = (f_{k-m_k}, \ldots, f_k)$.
Determine $\alpha^k = \{\alpha_1^k, \alpha_2^k, \ldots, \alpha_{m_k}^k\}$ according to equation (38).
Set $\lambda^{k+1} = \sum_{i=0}^{m_k} G(\lambda_{k+m_k+i})$ according to equation (39).
Step 4 (Check stopping criterion)
If $\mathrm{abs}(f_k) < \epsilon$, then set x_i according to equations (30) and (33) and STOP. Otherwise, set $k = k + 1$ and return to **Step 2**.

Algorithm 3: FPA2-Anderson algorithm.

5.2. Aitken Acceleration. As in [9], Aitken acceleration's idea is to change the relaxation factor (and, thus, the size of the iteration step) based on the information from the previous iteration.

Following [10], let us consider a sequence of scalars $\{\lambda_k\}_{k=0}^{\infty}$ that converges linearly to its fixed-point $\widehat{\lambda}$, which implies that for a large k:

$$\frac{\widehat{\lambda} - \lambda_{k+1}}{\widehat{\lambda} - \lambda_k} \approx \frac{\widehat{\lambda} - \lambda_{k+2}}{\widehat{\lambda} - \lambda_{k+1}}. \tag{40}$$

Below, we rearrange equation (40) to give a formula predicting the fixed point used as the subsequent iterate.

$$\lambda_{k+1} = \lambda_k - \frac{(\lambda_{k+1} - \lambda_k)^2}{\lambda_{k+2} - 2\lambda_{k+1} + \lambda_k}. \tag{41}$$

According to equation (41), we incorporate the Aitken approach into the FPA2 Algorithm 4.

6. Numerical Experiments

This section presents several numerical experiments using the FPA and FPA2 algorithms. The proposed algorithm is very simple and can be used to solve different forms of the quadratic convex separable knapsack problem.

We split our experiments into three subsections described as follows. In Subsection 6.1, we used randomly generated problems to compare the FPA2 and FPA algorithms with state-of-the-art solvers, the accelerated FPA2, and some root-finding methods. Then, we show the performance profile of the computational time for all algorithms presented. In Subsection 6.2, we solve the problem of finding the lowest risk portfolio. In Subsection 6.3, we perform the proposed algorithms with a continuous relaxation of the sensor placement problem presented in [13].

We implemented the methods in C and the compiler used was gcc 12.2.0 with optimization flags march=native -O3 -fast-math. All the experiments were performed on a Desktop with an Intel Core i5-9400 CPU (2.9 GHz). The

Step 0 (Initialization)
Set $k = 0$. Let $\lambda^k \in \mathbb{R}$ according to [2, 6, 12].
Step 1 (Calculating dual bounds)
For $i \in \{1, 2, \ldots, n\}$,
Compute $[\mathrm{LR}_i, \mathrm{UR}_i] = [a_i'/b_i, a_i' - p_i u_i'/b_i]$.
Step 2 (Calculating fixed-point sums and defining λ_1^k)
Compute SLU_1^k, SLE_{a1}^k and SLE_{b1}^k using λ^k according to FPA2 algorithm.
Compute $\lambda_1^k = \mathrm{SLU}_1^k + \mathrm{SLE}_{a1}^k - c/\mathrm{SLE}_{b1}^k$.
Step 3 (Check stopping criterion)
If $\mathrm{abs}(\lambda_1^k - \lambda^k) < \epsilon$, then set x_i according to equations (30) and (33) and STOP.
Step 4 (Calculating fixed-point sums and defining λ_2^k)
Compute SLU_2^k, SLE_{a2}^k and SLE_{b2}^k using λ_1^k according to FPA2 algorithm.
Compute $\lambda_2^k = \mathrm{SLU}_2^k + \mathrm{SLE}_{a2}^k - c/\mathrm{SLE}_{b2}^k$.
Step 5 (Calculating Aitken acceleration procedure)
Compute $\lambda^{k+1} = \lambda^k - (\lambda_1^k - \lambda^k)^2/\lambda_2^k - 2\lambda_1^k + \lambda^k$.
Step 7 (Check stopping criterion)
If $\mathrm{abs}(\lambda^{k+1} - \lambda_2^k) < \epsilon$, then set x_i according to equations (30) and (33) and STOP. Otherwise, set $k = k + 1$ and return to **Step 2**.

Algorithm 4: FPA2-Aitken algorithm.

computer has 16 GB of memory and runs Ubuntu 20.04.6 64 bit .

The tables in the following subsections have an error column corresponding to the number of failed experiments. An experiment that includes one of the following items is considered a failure:

(i) Reach the maximum number of 100 iterations.

(ii) The relative residual is not small enough $(1e^{-8})$.

(iii) The optimal value, when the problem is viewed as a D-projection, is not approximately equal to the other solvers.

6.1. Random Generated Problems. In this section, we generate random problems into medium and large problems, e.g., dimensions from $n = 500,000$ to $n = 50,000,000$. As in [2, 16], the problems were divided into four classes:

(1) Uncorrelated: $a_i, b_i, p_i \in [10, 25]$.

(2) Weakly correlated: $b_i \in [10, 25]$, $a_i, p_i \in [b_i - 5, b_i + 5]$.

(3) Strongly correlated: $b_i \in [10, 25], a_i = p_i = b_i + 5$.

(4) Flow: $p_1 = 1$, $p_n = 10^4$, $p_i \sim U[d_1, d_n]$ for $i = 2, \ldots, n-1$, and $a_i \sim U[-1000, 1000]$, $b_i = 1$, $l_i = 0$, $u_i \sim U[0, 1000]$ all for $[i = 1, \ldots, n]$ while r was selected uniformly in $[b^T l, b^T u]$.

Furthermore, for the problem classes 1, 2, and 3, l_i, u_i were chosen uniformly as in [2], $i \in N$ and $c \in [b^T l, b^T u]$.

6.1.1. Comparison with State-of-the-Art Algorithms. In the first experiment, we consider the Newton-based method [2], variable fixing [12], secant-based method [14], and median search [17].

From Tables 1–4, we can see the results of FPA2, FPA, Newton, secant, variable fixing, and median search algorithms in milliseconds over 50 randomly generated tests for each dimension and class. Each random test was repeated 20 times in a loop to obtain a reliable estimate for the running time. We report the mean time of each random test. The stopping criterion used for the algorithms FPA2 and FPA is according to the criteria

Table 1: Uncorrelated tests for state-of-the-art algorithms.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
FPA2							
500000	5.3	8	4	0.010	0.018	0.009	0
1000000	5.5	10	4	0.021	0.032	0.018	0
10000000	5.4	10	4	0.354	0.406	0.324	0
50000000	5.3	8	3	1.774	1.964	1.556	0
Newton							
500000	5.3	8	4	0.018	0.023	0.015	0
1000000	5.5	10	4	0.038	0.047	0.030	0
10000000	5.2	10	4	0.434	0.535	0.346	0
50000000	5.2	7	3	2.181	2.649	1.555	0
Variable fixing							
500000	8.8	11	8	0.022	0.025	0.021	0
1000000	9.0	12	7	0.047	0.053	0.041	0
10000000	9.3	12	8	0.539	0.622	0.468	0
50000000	9.5	11	8	2.718	3.065	2.377	0
FPA							
500000	5.3	8	4	0.011	0.020	0.010	0
1000000	5.5	10	4	0.023	0.036	0.020	0
10000000	5.4	10	4	0.416	0.482	0.377	0
50000000	5.3	8	3	2.078	2.360	1.904	0
Secant							
500000	8.7	14	7	0.020	0.028	0.014	0
1000000	8.6	13	6	0.041	0.059	0.029	0
10000000	8.5	14	7	0.443	0.658	0.285	0
50000000	8.5	12	6	2.323	3.254	1.426	0
Median search							
500000	19.9	20	19	0.037	0.040	0.032	0
1000000	20.8	21	20	0.077	0.085	0.067	0
10000000	24.3	25	24	0.895	1.010	0.785	0
50000000	26.7	27	26	4.562	5.059	4.189	0

Table 2: Weakly correlated test for state-of-the-art algorithms.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
FPA2							
500000	5.2	8	4	0.010	0.018	0.009	0
1000000	5.2	9	4	0.021	0.038	0.018	0
10000000	5.2	10	3	0.360	0.407	0.321	0
50000000	5.3	7	4	1.786	1.951	1.649	0
Newton							
500000	5.1	8	4	0.019	0.024	0.015	0
1000000	5.2	9	4	0.037	0.050	0.032	0
10000000	5.1	10	3	0.419	0.501	0.318	0
50000000	5.1	7	3	2.105	2.476	1.721	0
Variable fixing							
500000	8.2	11	7	0.022	0.025	0.019	0
1000000	8.5	11	7	0.046	0.053	0.040	0
10000000	9.0	12	8	0.534	0.589	0.483	0
50000000	9.1	10	8	2.610	2.911	2.372	0
FPA							
500000	5.2	8	4	0.011	0.021	0.010	0
1000000	5.2	9	4	0.023	0.043	0.020	0
10000000	5.2	10	3	0.419	0.483	0.369	0
50000000	5.3	7	4	2.085	2.434	1.927	0
Secant							
500000	8.5	13	7	0.021	0.037	0.012	0
1000000	8.4	13	6	0.039	0.066	0.023	0
10000000	8.3	14	6	0.417	0.658	0.235	0
50000000	8.4	12	6	2.236	3.186	1.094	0
Median search							
500000	19.9	20	19	0.039	0.049	0.033	0
1000000	20.9	21	20	0.078	0.098	0.066	0
10000000	24.2	25	24	0.916	1.049	0.787	0
50000000	26.8	27	26	4.478	4.891	4.031	0

Table 3: Correlated test for state-of-the-art algorithms.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
FPA2							
500000	4.9	8	3	0.010	0.017	0.008	0
1000000	5.2	8	4	0.020	0.036	0.018	0
10000000	5.2	10	4	0.356	0.421	0.329	0
50000000	5.0	9	4	1.730	1.968	1.617	0
Newton							
500000	4.9	8	3	0.017	0.021	0.013	0
1000000	5.1	8	4	0.036	0.045	0.031	0
10000000	5.0	10	4	0.407	0.494	0.359	0
50000000	4.9	9	4	1.961	2.345	1.757	0
Variable fixing							
500000	8.2	10	7	0.021	0.024	0.019	0
1000000	8.4	10	7	0.045	0.049	0.040	0
10000000	8.9	12	8	0.517	0.573	0.482	0
50000000	9.0	12	8	2.542	2.695	2.338	0
FPA							
500000	4.9	8	3	0.011	0.020	0.009	0
1000000	5.2	8	4	0.023	0.041	0.020	0
10000000	5.2	10	4	0.418	0.507	0.383	0
50000000	5.0	9	4	2.016	2.305	1.883	0
Secant							
500000	8.1	12	6	0.018	0.032	0.012	0
1000000	8.5	11	7	0.037	0.064	0.025	0
10000000	8.3	12	6	0.382	0.637	0.239	0
50000000	7.9	11	6	1.927	3.082	1.016	0
Median search							
500000	19.9	20	19	0.037	0.040	0.034	0
1000000	21.0	21	20	0.075	0.082	0.068	0
10000000	24.3	25	24	0.896	1.042	0.790	0
50000000	26.7	27	26	4.375	4.570	3.893	0

$$\operatorname{abs}(\lambda_k - \lambda_{k-1}) \leq \epsilon, \tag{42}$$

where ϵ is a small positive number greater than 0 and its value was chosen as $\epsilon = 10e^{-12}$ as in [2].

The results show that the computational time of the FPA2 algorithm was superior in all experiments than the other state-of-the-art methods. The FPA algorithm was superior to the other methods only for weakly correlated and uncorrelated problems with a large number of variables, e.g., $n = 10{,}000{,}000$ and $n = 50{,}000{,}000$. For the other problems, the FPA does not perform better than the Newton algorithm.

In the largest ($n = 50{,}000{,}000$) test, Tables 1–4 show the following results:

(i) For the uncorrelated and weakly correlated problems, the FPA2 algorithm was about 14%, 18%, 20%, 34%, and 154% faster than FPA, Newton, secant, variable fixing, and median search algorithms, respectively.

(ii) For the correlated problems, the FPA2 algorithm was about 14%, 11%, 10%, 31%, and 152% faster than FPA, Newton, secant, variable fixing, and median search algorithms, respectively.

(iii) For the flow problems, the FPA2 algorithm was about 15%, 8%, 10%, 2%, and 100% faster than FPA, Newton, secant, variable fixing, and median search algorithms, respectively.

The results show yet that all algorithms solved all problems correctly.

6.1.2. Comparison with Accelerated Algorithms. In this subsection, we compare the performance of the FPA2 and FPA algorithms with two new versions of the accelerated FPA2 algorithm. We incorporated the Aitken and Anderson acceleration approach in the FPA2 algorithm since the FPA2 presented a better performance in Subsection 6.1.

As in [9, 10], the fixed-point acceleration approach performs better than the original fixed-point algorithm. The numerical experiments in [9] show that for a specific engineering problem, Anderson and Aitken's acceleration has similar performance depending on the parameters defined in the algorithm. In [10], fixed-point accelerations are applied in different scenarios. The default acceleration approach for the solver proposed in [10] is Anderson's acceleration since it shows better results in the numerical experiments.

Anderson's acceleration-based algorithm, FPA2-Anderson, presents slight improvement compared with the FPA2 algorithm for the uncorrelated and correlated problems when the problem size is larger. For the class of

Table 4: Flow test for state-of-the-art algorithms.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
FPA2							
500000	6.2	8	5	0.010	0.018	0.008	0
1000000	6.5	9	5	0.021	0.040	0.017	0
10000000	6.4	11	4	0.372	0.446	0.311	0
50000000	6.5	9	5	1.813	2.027	1.628	0
Newton							
500000	6.1	8	4	0.017	0.022	0.010	0
1000000	6.5	9	5	0.036	0.048	0.025	0
10000000	6.1	10	4	0.403	0.507	0.258	0
50000000	6.2	9	5	1.978	2.604	1.542	0
Variable fixing							
500000	6.2	8	5	0.015	0.021	0.010	0
1000000	6.5	9	5	0.033	0.044	0.022	0
10000000	6.5	11	4	0.376	0.467	0.223	0
50000000	6.6	9	5	1.860	2.395	1.340	0
FPA							
500000	6.2	8	5	0.011	0.021	0.009	0
1000000	6.5	9	5	0.024	0.046	0.020	0
10000000	6.4	11	4	0.438	0.531	0.369	0
50000000	6.5	9	5	2.135	2.351	1.925	0
Secant							
500000	14.4	16	9	0.018	0.022	0.013	0
1000000	14.6	19	11	0.040	0.052	0.029	0
10000000	14.3	21	10	0.425	0.791	0.283	0
50000000	14.2	17	9	2.026	3.088	1.511	0
Median search							
500000	20.0	20	19	0.030	0.034	0.028	0
1000000	20.9	21	20	0.061	0.068	0.057	0
10000000	24.3	25	24	0.750	0.825	0.715	0
50000000	26.6	27	26	3.657	4.012	3.507	0

problems weakly correlated and flow, the FPA2 performed better than the accelerated algorithms. Furthermore, the FPA2-Anderson could only solve some of the problems of each class correctly. The FPA2-Aitken algorithm correctly solves almost all the problems, as visualized in the column Error. In our experiments, presented in Tables 5–8, in comparison with computation time, Anderson's acceleration performed better than the Aitken acceleration.

6.1.3. Comparison with Root-Finding Algorithms. We also compare our fixed-point approach with some root-finding algorithms, as in [3]. Tables 9–11 show the result for the secant, bisection, and regula falsi methods, respectively. All pseudocodes for these methods can be found in [3].

In this subsection, we do not perform the algorithms with the flow problem because the root-finding algorithms need many iterations to reach good results with the established ϵ value in the stop criterion. Furthermore, we can see throughout column Error that the secant, regula falsi, and bisection algorithms did not solve some problems well. The FPA2 algorithm reported here is the same as in Subsections 6.1.1 and 6.1.2, so the results are very similar.

We highlight the high superiority of the FPA2 algorithm compared to other popular root-finding algorithms.

Table 5: Uncorrelated test for accelerated algorithms.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
FPA2-Aitken							
500000	2.5	6	2	0.017	0.022	0.016	1
1000000	2.4	4	1	0.035	0.039	0.031	1
10000000	2.3	5	2	0.363	0.474	0.333	1
50000000	2.5	4	2	1.749	1.957	1.612	0
FPA2							
500000	5.5	8	4	0.016	0.018	0.015	0
1000000	5.4	10	4	0.034	0.040	0.031	0
10000000	5.3	10	4	0.355	0.406	0.325	0
50000000	5.3	8	4	1.681	1.864	1.585	0
FPA2-Anderson							
500000	4.4	16	2	0.017	0.026	0.015	2
1000000	4.3	11	2	0.036	0.046	0.031	1
10000000	3.3	5	2	0.358	0.417	0.323	9
50000000	3.2	5	2	1.676	1.839	1.568	10
FPA							
500000	5.5	8	4	0.019	0.022	0.018	0
1000000	5.4	10	4	0.040	0.047	0.037	0
10000000	5.3	10	4	0.418	0.487	0.378	0
50000000	5.3	8	4	1.982	2.206	1.855	0

Table 6: Weakly correlated test for accelerated algorithms.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
FPA2-Aitken							
500000	2.3	4	1	0.019	0.022	0.017	0
1000000	2.2	5	1	0.037	0.048	0.031	0
10000000	2.1	5	1	0.368	0.456	0.318	0
50000000	2.1	3	1	1.820	2.159	1.624	0
FPA2							
500000	5.2	8	4	0.019	0.021	0.018	0
1000000	5.2	9	4	0.036	0.042	0.032	0
10000000	5.3	10	4	0.362	0.416	0.329	0
50000000	5.2	7	4	1.841	2.045	1.642	0
FPA2-Anderson							
500000	5.3	57	2	0.021	0.064	0.017	0
1000000	3.7	8	2	0.037	0.042	0.031	1
10000000	3.2	5	2	0.367	0.427	0.325	7
50000000	3.5	5	2	1.848	2.056	1.633	5
FPA							
500000	5.2	8	4	0.023	0.026	0.021	0
1000000	5.2	9	4	0.042	0.050	0.037	0
10000000	5.3	10	4	0.429	0.496	0.389	0
50000000	5.2	7	4	2.137	2.470	1.930	0

6.1.4. Performance Profile Analysis of the Algorithms. The approach adopted to analyze and compare the performance profile of the algorithms developed in this work was proposed in [18]. The authors created it intending to facilitate the visualization and interpretation of the results obtained in experiments, comparing a set of algorithms to identify the one with the best performance applied to a set of problems. The method considers a set P of test problems p_j, with $j = 1, 2, \ldots, n_p$, a set of algorithms a_i, with $i = 1, 2, \ldots, n_a$, and a performance metric $t_{p,a}$ (computation time, an average of objective function values, and others).

Table 7: Correlated test for accelerated algorithms.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
FPA2-Aitken							
500000	2.2	4	1	0.017	0.019	0.015	0
1000000	2.3	4	1	0.036	0.042	0.031	0
10000000	2.2	4	1	0.375	0.440	0.317	0
50000000	2.1	4	1	1.834	2.001	1.646	0
FPA2							
500000	5.0	8	3	0.016	0.018	0.015	0
1000000	5.3	8	4	0.035	0.042	0.032	0
10000000	5.2	10	4	0.367	0.459	0.329	0
50000000	5.1	9	4	1.807	1.983	1.657	0
FPA2-Anderson							
500000	3.5	10	1	0.017	0.021	0.012	0
1000000	4.0	9	2	0.037	0.043	0.032	0
10000000	3.3	5	2	0.371	0.412	0.326	5
50000000	3.2	5	2	1.802	2.102	1.609	5
FPA							
500000	5.0	8	3	0.020	0.021	0.018	0
1000000	5.3	8	4	0.042	0.051	0.037	0
10000000	5.2	10	4	0.434	0.547	0.388	0
50000000	5.1	9	4	2.115	2.385	1.918	0

Table 8: Flow test for accelerated algorithms.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
FPA2-Aitken							
500000	3.2	6	2	0.019	0.022	0.015	0
1000000	3.5	6	2	0.039	0.050	0.032	2
10000000	3.2	6	2	0.404	0.492	0.334	0
50000000	3.2	5	2	1.978	2.266	1.690	2
FPA2							
500000	6.2	8	5	0.018	0.021	0.015	0
1000000	6.6	9	5	0.036	0.043	0.031	0
10000000	6.3	8	4	0.390	0.433	0.329	0
50000000	6.5	9	5	1.871	2.109	1.675	0
FPA2-Anderson							
500000	6.2	24	3	0.019	0.034	0.015	2
1000000	6.5	23	3	0.040	0.074	0.032	9
10000000	4.2	5	2	0.380	0.422	0.321	24
50000000	4.6	5	3	1.874	2.004	1.678	23
FPA							
500000	6.2	8	5	0.021	0.027	0.018	0
1000000	6.6	9	5	0.043	0.054	0.037	0
10000000	6.3	8	4	0.460	0.517	0.379	0
50000000	6.5	9	5	2.197	2.502	1.953	0

Table 9: Uncorrelated test for root-finding algorithms.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
FPA2							
500000	5.3	8	4	0.016	0.018	0.015	0
1000000	5.5	10	4	0.034	0.039	0.031	0
10000000	5.4	10	4	0.347	0.408	0.323	0
50000000	5.3	8	3	1.693	1.860	1.529	0
Regula falsi							
500000	14.3	78	6	0.094	0.220	0.054	8
1000000	18.2	90	5	0.200	0.515	0.106	6
10000000	11.9	32	5	1.791	2.721	1.199	4
50000000	12.9	51	6	9.064	17.161	4.563	1
Secant							
500000	8.4	13	3	0.066	0.094	0.043	15
1000000	8.6	14	5	0.140	0.200	0.098	8
10000000	9.0	13	6	1.499	2.007	1.007	16
50000000	8.7	12	5	7.126	10.222	4.061	10
Bisection							
500000	49.8	53	45	0.134	0.201	0.065	6
1000000	49.4	53	43	0.274	0.409	0.135	4
10000000	49.8	53	47	2.851	4.021	1.538	1
50000000	50.0	53	46	14.283	20.306	7.665	1

Table 10: Weakly correlated test for root-finding algorithms.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
FPA2							
500000	5.2	8	4	0.016	0.017	0.015	0
1000000	5.2	9	4	0.033	0.038	0.031	0
10000000	5.2	10	3	0.343	0.411	0.306	0
50000000	5.3	7	4	1.685	1.829	1.574	0
Regula falsi							
500000	14.9	94	4	0.088	0.240	0.048	5
1000000	12.8	39	6	0.166	0.277	0.109	6
10000000	12.7	72	6	1.757	4.307	1.043	5
50000000	12.3	29	5	9.087	15.780	5.568	7
Secant							
500000	8.7	15	5	0.066	0.096	0.047	13
1000000	9.1	16	6	0.134	0.212	0.100	13
10000000	9.1	12	7	1.413	1.814	1.071	14
50000000	9.0	11	6	7.054	10.166	5.133	18
Bisection							
500000	50.1	53	43	0.128	0.196	0.066	4
1000000	49.5	53	46	0.254	0.408	0.138	4
10000000	49.7	53	45	2.760	4.080	1.549	3
50000000	50.3	53	46	14.097	20.289	8.536	7

The performance ratio (always greater than or equal to 1) is defined as

$$r_{p,a} = \frac{t_{p,a}}{\min\left(t_{p,a}\colon a \in A\right)}. \tag{43}$$

The algorithm performance profile is given by

$$\rho_a(\tau) = \frac{1}{n_p} p \in P\colon r_{p,a} \leq \tau, \tag{44}$$

where $\rho_a(\tau)$ is the fraction of problems solved by the algorithm with performance within a factor of the best performance obtained, considering all algorithms.

In Figure 1, we display the graphs regarding the performance profile of [18] concerning computational time for the problems correlated, uncorrelated, weakly correlated, and flow with size $n = 2,000,000$. In this experiment, we use the same data to generate the tables presented in Subsections 6.1 and 6.2.

Table 11: Correlated test for root-finding algorithms.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
FPA2							
500000	4.9	8	3	0.015	0.017	0.014	0
1000000	5.2	8	4	0.033	0.036	0.031	0
10000000	5.2	10	4	0.343	0.410	0.322	0
50000000	5.0	9	4	1.633	1.878	1.544	0
Regula falsi							
500000	11.2	40	6	0.086	0.132	0.059	5
1000000	11.9	77	4	0.169	0.443	0.113	6
10000000	13.3	96	6	1.861	6.058	1.224	11
50000000	11.5	66	4	9.078	21.548	5.451	5
Secant							
500000	8.7	12	6	0.072	0.109	0.058	15
1000000	8.6	11	6	0.143	0.201	0.113	12
10000000	8.5	12	6	1.466	2.178	1.172	17
50000000	8.6	15	6	7.598	10.430	5.978	13
Bisection							
500000	50.1	53	45	0.135	0.206	0.056	4
1000000	49.9	53	45	0.258	0.393	0.186	6
10000000	49.8	53	42	2.692	4.051	1.162	8
50000000	50.5	53	44	14.889	20.946	6.336	4

Figure 1 shows that although the average time of the FPA2-Aitken algorithm is faster than all other algorithms, for some of the 50 samples generated in each experiment, the other algorithms performed better. Below we describe Figure 1 based on the individual results of the 50 generated samples for each problem type.

(i) Uncorrelated: FPA2-Aitken and Newton solved about 46% and 30% of the samples faster, respectively, while the variable fixing method solved about 20% of the samples faster;

(ii) Correlated: the variable fixing method and FPA2-Aitken solved about 54% and 34% of the samples faster, respectively. Newton and FPA2-Anderson together solved about 12% of the samples faster.

(iii) Weakly correlated: FPA2-Aitken and variable fixing method solved about 52% and 36% of the samples faster, respectively, while Newton, FPA2-Anderson, secant, and median search together summed solved about 12% of the problems faster.

(iv) Flow: secant and FPA2-Aitken solved about 60% and 40% of the samples faster, respectively.

The results presented in Subsections 6.1–6.3 and 6.1.4 reinforce the competitiveness of the method proposed in this article. Simplicity and easy implementation contribute to placing the FPA2 and FPA algorithms, as well as their accelerated versions, as excellent options for solving problem (1).

6.2. The Portfolio Optimization. In this subsection, following the experiments presented in [19], we apply the FPA algorithm to the portfolio optimization problem based on the mean-variance model [20]. Knowing that there is a trade-off between reward and risk in investment portfolios, the investor must be willing to tolerate risk to obtain ever-increasing returns.

An investment portfolio is defined by the vector $x = (x_1, x_2, \ldots, x_n)^T$, where x_i denotes the proportion of the investment to be invested in asset i. Assuming that all available assets are invested, then the problem should satisfy the constraints $\sum_{i=1}^{n} x_i = 1$ and $x_i \geq 0$, for $i = 1, 2, \ldots, n$. So, the portfolio optimization problem can be written as

$$\begin{aligned} \text{Minimize} \quad & f(x) = x^T Q x, \\ & e^x = 1, \\ s.t. \quad & x \geq 0, \end{aligned} \tag{45}$$

where $Q \in \mathbb{R}^{n \times n}$ is the covariance matrix and $e = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$.

As it can be seen, model (45) is a special case of problem (1) with $f(x) = x^T Q x$ and $Q \in \mathbb{R}^{n \times n}$, $a = (0, 0, \ldots, 0)^T$, $b = 1 = (1, 1, \ldots, 1)^T$, $c = 1$, and $l = (0, 0, \ldots, 0)^T$. As $e^T x = 1$ and $x \geq 0$, these constraints are equivalent to $e^T x = 1$ and $0 \leq x_i \leq 1$, for all $i = 1, 2, \ldots, n$, and thus $u = (1, 1, \ldots, 1)^T$.

Since the FPA algorithm solves a separable problem, we must reformulate model (45). In this case, we can use a framework proposed in [21] to solve nonseparable optimization problems in the form of problem (45). This framework was first proposed for training SVM, but it can also be applied to portfolio optimization.

The framework comprises two main stages: the first consists of approximating the objective function of the main problem into a separable objective function with a diagonal Hessian matrix, and the second stage consists of solving the subproblem with the FPA algorithm.

In [21], the authors use a separable quadratic approximation, defined firstly in [22], to obtain a diagonal approximation of f at the point x^k, named $\overline{f}_k(x)$. The function $\overline{f}_k$ has the following formula:

$$\begin{aligned} \overline{f}_k(x) = {} & \frac{1}{2}\left(x^{k-1} - x^k\right)^T Z_k \left(x^{k-1} - x^k\right) \\ & + \nabla f\left(x^k\right)^T \left(x^{k-1} - x^k\right) + f\left(x^k\right), \end{aligned} \tag{46}$$

where $Z_k = \operatorname{diag}(z_k, z_k, \ldots, z_k) = z_k I$ and k is the current iteration. Snyman and Hay [22] declare that forcing $\overline{f}_k(x^{k-1}) = f(x^{k-1})$, it is possible to obtain z_{k+1} as follows:

$$z_{k+1} = \frac{2\left[f\left(x^{k-1}\right) - f\left(x^k\right) - \nabla f\left(x^k\right)^T \left(x^{k-1} - x^k\right)\right]}{\left\|x^{k-1} - x^k\right\|^2}. \tag{47}$$

As a result of the separable approximation in the first stage of the framework and after reformulating objective function (46) to the format we are looking for in (45), the following problem is obtained:

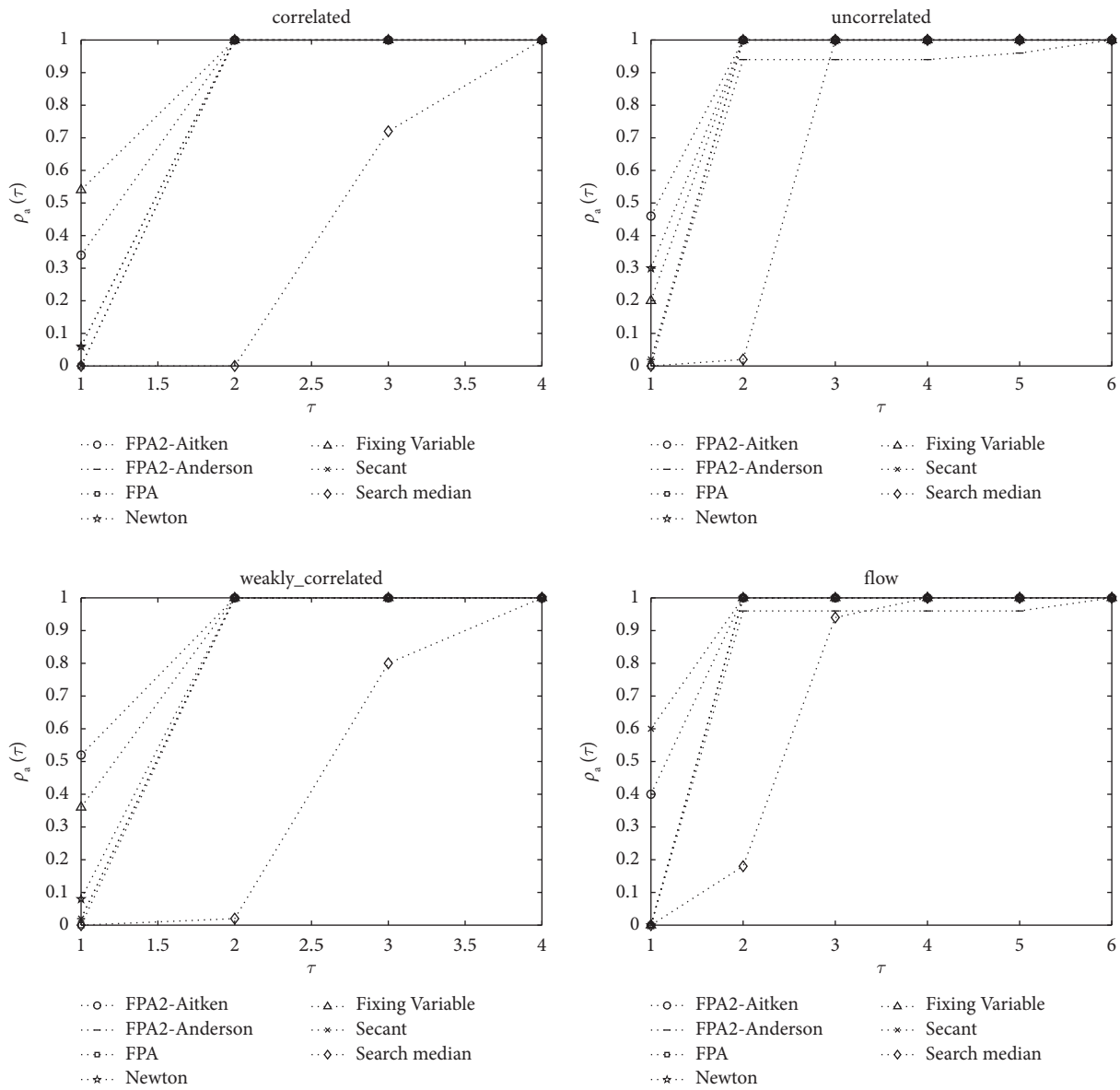

Figure 1: Performance profile of the algorithms applied to the correlated, uncorrelated, weakly correlated, and flow test problems with size $n = 2{,}000{,}000$.

$$\begin{aligned} \text{Minimize} \quad & \overline{f}(x) = \frac{1}{2} x^T Z_k x, \\ & e^T x = 1, \\ s.t \quad & 0 \le x \le e. \end{aligned} \tag{48}$$

Thus, the FPA algorithm solves problem (48) in each iteration of the framework until the stopping criterion in the FPA algorithm is achieved.

We use asset data from the Brazilian stock exchange to apply our algorithm to model (48). More specifically, we searched the returns of the shares that were part of the Ibovespa Index during the trading sessions of five years, namely, from 01-01-2020 to 30-12-2022. For that, we used Yahoo Finance and 58 assets, all with all returns in the period. The period considered produced 745 trading sessions on the stock exchange, and the dimension of the covariance matrix is 58 lines by 58 columns. Therefore, a total of 58 assets will compose the portfolio, listed in Table 12.

In Figure 2, MR is the optimal portfolio determined by our algorithm applied to model (48), which has a return of 0.0502 and risk of 1.3818 by day. The other points are portfolios investing 100% in each asset. The portfolio WEGE3 is determined by investing 100% in company WEG; this is the portfolio with the highest return, which is 0.1497. The portfolio PETR4 is determined by investing 100% in company PETROBRAS, and it has the lowest return among all portfolios. Besides, it is negative, and its value is −0.1782.

As in [19], we also consider the diversification of the assets. All portfolios formed by investing 100% in any of the individual assets have a greater risk than the optimal

Table 12: Assets in the portfolio and their names.

Order	Tickers	Name
1	ABEV3	AMBEV S/A
2	B3SA3	B3
3	BBAS3	BRASIL
4	BBDC3	BRADESCO
5	BBDC4	BRADESCO
6	BBSE3	BBSEGURIDADE
7	BEEF3	MINERVA
8	BRAP4	BRADESPAR
9	BRFS3	RF SA
10	BRKM5	BRASKEM
11	CCRO3	CCR SA
12	CIEL3	CIELO
13	CMIG4	CEMIG
14	COGN3	COGNA ON
15	CPFE3	CPFL ENERGIA
16	CSAN3	COSAN
17	CSNA3	SID NACIONAL
18	CVCB3	CVC BRASIL
19	CYRE3	CYRELA REALT
20	ECOR3	ECORODOVIAS
21	EGIE3	ENGIE BRASIL
22	ELET3	ELETROBRAS
23	ELET6	ELETROBRAS
24	EMBR3	EMBRAER
25	ENBR3	ENERGIAS BR
26	ENGI11	ENERGISA
27	EQTL3	EQUATORIAL
28	FLRY3	FLEURY
29	GGBR4	GERDAU
30	GOAU4	GERDAU MET
31	GOLL4	GOL
32	HYPE3	HYPERA
33	ITSA4	ITAUSA
34	ITUB4	ITAUUNIBANCO
35	JBSS3	JBS
36	KLBN11	KLABIN S/A
37	LREN3	LOJAS RENNER
38	MGLU3	MAGAZINE LUIZA
39	MRFG3	MARFRIG
40	MRVE3	MRV
41	MULT3	MULTIPLAN
42	QUAL3	QUALICORP
43	PCAR3	P.ACUCAR-CBD
44	PETR3	PETROBRAS
45	PETR4	PETROBRAS
46	RADL3	RAIADROGASIL
47	RAIL3	RUMO S.A.
48	RENT3	LOCALIZA
49	SANB11	SANTANDER BR
50	SBSP3	SABESP
51	SUZB3	SUZANO S.A.
52	TAEE11	TAESA
53	TOTS3	TOTVS
54	UGPA3	ULTRAPAR
55	USIM5	USIMINAS
56	VALE3	VALE
57	WEGE3	WEG
58	YDUQ3	YDUQS PART

portfolio, which is formed by combining several assets, as in the MR portfolio. We highlight this in Table 13 , which shows the optimal MR portfolio.

In Table 13, the order is the order of the assets, prop is the proportion of the assets in the MR portfolio, $\overline{R}_i$ is the expected return of asset i, and σ_i is the risk of asset i. For example, order 1 is asset 1, which, according to Table 12, is AMBEV S/A. The proportion of this asset to the optimal MR portfolio is 1.2243, whose expected return and risk are 0.0029 and 4.8227, respectively, similar to the other assets in orders 2 to 58. In addition, we highlight the assets WEG in order 57 and PETROBRAS in order 45, with a proportion equal to 0 in the optimal MR portfolio.

In the following portfolio formulation, we incorporate a risk tolerance parameter denoted by ρ, and model (45) becomes

$$\begin{aligned} \text{Minimize} \quad & f(x) = x^T Q x - \frac{1}{\rho}\overline{R}^T x, \\ & e^T x = 1, \\ s.t \quad & \\ & 0 \le x \le e, \end{aligned} \tag{49}$$

where $\overline{R} = (\overline{R}_1, \overline{R}_2, \ldots, \overline{R}_n)^T$ is the vector of the expected returns of the n assets and $\rho \in (0, \infty)$ is the preference of the individual investor which is also known as a risk aversion parameter.

To use the FPA algorithm in problem (49), we need to apply the framework as shown previously. In this case, we reformulate the objective function (46) according to the objective function in (49), resulting in the following subproblem:

$$\begin{aligned} \text{Minimize} \quad & \overline{f}(x) = \frac{1}{2}x^T Z_k x - \frac{1}{\rho}\left[-\nabla f\left(x^k\right) + Z_k x^k\right]^T x, \\ & e^T x = 1, \\ \text{s.t.} \quad & \\ & 0 \le x \le e. \end{aligned} \tag{50}$$

Similar to problem (48), the FPA algorithm solves problem (50) in each iteration of the framework until the stopping criterion in the FPA algorithm is achieved.

We run our algorithm to solve problem (50) with the same data used in the first example, and the assets are given in Table 12. For each fixed value of ρ in problem (50), we will have an optimal portfolio determined by the algorithm. If an investor is risk averse, he will choose a considerable value for ρ, meaning he wants to minimize the risk. On the other hand, if the investor is more tolerant of risk, he will choose a small ρ, giving more weight to the return.

Figure 3 shows 200 portfolios determined by our algorithm varying ρ from 0.5 to 100 by steps of the size of 0.5, that is, the first portfolio was determined by taking $\rho = 0.5$,

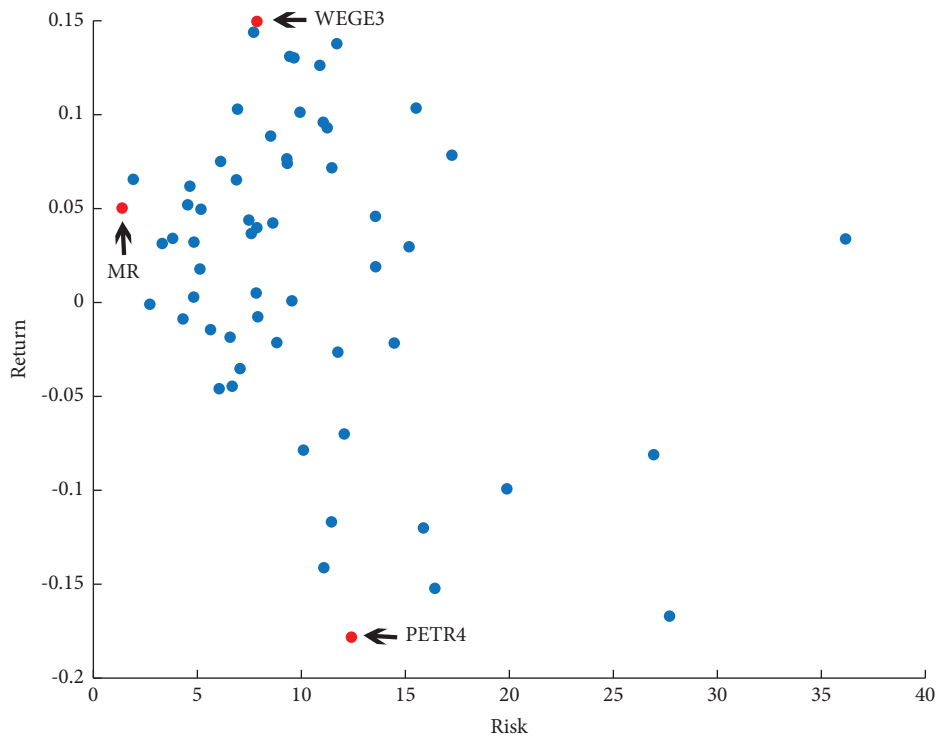

Figure 2: Ibovespa stocks and portfolios.

which is the portfolio in the right corner of Figure 3 and it has expected return of 0.1473 and the risk of 5.1151. The next one is determined by taking $\rho = 1.0$ until $\rho = 100$, which is the portfolio in the left corner of Figure 3 with an expected return of 0.0538 and the risk of 1.3854. It is almost identical to the MR portfolio determined by problem (48).

6.3. Sensor Placement Problem. In this section, we address the FPA algorithm to a type of traffic problem named sensor placement problem. The model used in our experiments is addressed to optimization problems according to [23], where there is a single commodity demand that has to be satisfied, a set of potential resources, and a fixed activation cost for each resource, and the congestion heavily influences the cost of a resource. Such model is formulated as a mixed integer nonlinear programming problem (MINLP). Following [13], we consider the problem of optimally placing a set $N = \{1, \ldots, n\}$ of sensors to cover a given area, where deploying one sensor has a fixed cost plus a cost that is quadratic in the radius of the surface covered. The problem can be written as

$$\begin{aligned} &\min \sum_{i \in N} d x_i^2 + \sum_{i \in N} y_i a_i, \\ &s.t. \sum_{i \in N} x_i = 1, \\ &0 \le x_i \le y_i, \forall i \in N, \\ &y_i \in \{0, 1\}, \forall i \in N, \end{aligned} \tag{51}$$

where x_i indicates the fraction of demand allocated to resource i and y_i is a binary variable indicating whether resource i is active ($x_i > 0 \Longrightarrow y_i = 1$) or not ($x_i = 0 \Longrightarrow y_i = 0$).

Now, in the continuous relaxation of problem (51), Reference [23] relaxes the integrality constraint on the y variables. Since we can assume $a_i > 0$ (for otherwise y_i can surely be fixed to 1), the "design" variables y_i can be "projected" onto x_i. The problem is now the following:

$$\begin{aligned} &\min \sum_{i \in N} d x_i^2 + \sum_{i \in N} \mathrm{a} x_i, \\ &s.t. \sum_{i \in N} x_i = 1, \\ &x_i \ge 0, \forall i \in N. \end{aligned} \tag{52}$$

Problem (52) is a representative case of the continuous nonlinear resource allocation problem that, in addition to the sensor placement problem, has many applications, see reference [5].

In [13, 23], some complexity issues were proved for problems (51) and (52). The instances of the sensor placement problem were generated with the generator freely available at https://groups.di.unipi.it/optimize/Data/RDR.html.

We generated five types of random problems based on their length: $n = 100$, $n = 1,000$, $n = 10,000$, $n = 100,000$ and $n = 1,000,000$.

Table 13: The optimal portfolio determined by our algorithm applied to problem (29).

Order	Prop	R_p	σ_p
1	1.2243	0.0029	4.8227
2	0.0000	0.0423	8.6283
3	0.0000	0.0050	7.8293
4	0.0000	−0.0446	6.6740
5	0.0000	−0.0352	7.0525
6	10.5258	0.0341	3.8142
7	0.0000	0.0765	9.3015
8	2.4055	0.1013	9.9341
9	0.0000	−0.1412	11.0816
10	0.0000	0.0784	17.2383
11	0.0000	−0.0213	8.8191
12	0.0000	0.0190	13.5723
13	0.0000	0.1030	6.9312
14	0.0000	−0.1522	16.4183
15	0.0000	0.0619	4.6405
16	0.0000	0.0439	7.4761
17	0.0000	0.1035	15.5176
18	0.0000	−0.1670	27.7026
19	0.0000	−0.0216	14.4657
20	0.0000	−0.1168	11.4480
21	13.4308	−0.0009	2.7134
22	0.0000	0.0960	11.0521
23	0.0000	0.0886	8.5247
24	0.0000	0.0297	15.1808
25	0.0000	0.0314	3.3106
26	0.0000	0.0178	5.1229
27	0.0000	0.0520	4.5364
28	0.0000	−0.0459	6.0490
29	0.0000	0.1303	9.6479
30	0.0000	0.1310	9.4398
31	0.0000	−0.0810	26.9398
32	0.0000	0.0751	6.1179
33	0.0000	−0.0088	4.3083
34	0.0000	−0.0145	5.6327
35	0.0000	0.0367	7.5923
36	2.6346	0.0496	5.1736
37	0.0000	−0.0786	10.0993
38	0.0000	−0.0992	19.8807
39	0.0000	0.0930	11.2444
40	0.0000	−0.0700	12.0619
41	0.0000	0.0009	9.5446
42	0.2759	0.0338	36.1665
43	0.0000	0.1378	11.7048
44	0.0000	0.1263	10.8888
45	0.0000	−0.1782	12.4063
46	7.9233	0.0322	4.8360
47	0.0000	−0.0076	7.9057
48	0.0000	0.0717	11.4650
49	0.0000	−0.0185	6.5744
50	0.0000	0.0398	7.8622
51	13.8906	0.0653	6.8785
52	47.6893	0.0656	1.9194
53	0.0000	0.0741	9.3250
54	0.0000	−0.0264	11.7549
55	0.0000	0.0459	13.5631
56	0.0000	0.1439	7.7052
57	0.0000	0.1497	7.8649
58	0.0000	−0.1201	15.8680
59		**0.0502**	**1.3818**

The bold values represents the expected return and risk of the optimal MR portfolio.

Table 14 presents the computational time necessary to solve each problem instance and the total number of iterations for the convergence of each method. As presented in Subsection 6.1.1, we choose to compare with the FPA and FPA2 the following methods: the Newton-based method [2], variable fixing [12], secant-based method [14], and median search [17].

We report the mean time of each random test. 10 randomly generated tests for each dimension were repeated 10 times in a loop to obtain a reliable estimate of the computational time, which is presented in milliseconds. The stopping criterion used for the algorithms is the same presented in Section 6.1.

The results in Table 14 show that all the algorithms solve the problem well except the secant method. For the generated problems with $n = 10,000$, the secant method could not solve four random tests before the maximum number of iterations. For the problems with $n = 100,000$ and $n = 1,000,000$, the secant method could not solve any random tests before the maximum number of iterations.

The FPA2 and FPA solved all the random problems with less computational time than the other algorithms and the number of iterations similar to Newton's method. The variable fixing method presented fewer iterations than FPA2, and the median search method presented fewer iterations than all algorithms.

7. Final Remarks

This article presents a straightforward root-finding numerical scheme for solving a quadratic convex separable knapsack problem based on a fixed-point algorithm studied in [3]. We named our algorithm as FPA. Then, we incorporate acceleration techniques as an alternative to improve the performance of the proposed algorithm.

To obtain better results in our experiments, we also reformulated the quadratic convex separable knapsack problem. Such reformulation allows for a new constraint format, and we named this new algorithm FPA2. We tested our algorithm by performing three different problems: randomly generated problems, portfolio optimization problem, and sensor placement problem. In the first experiment, we compared the FPA and FPA2 with some popular methods in the literature: Newton-based method [2], variable fixing [12], secant-based method [14], and median search [17]. For all random problems, the FPA2 obtained the best computational time in all problem sizes. The acceleration technique was applied to the FPA2 since it presented the best results in the previous experiments. Although the accelerated algorithms presented fewer iterations to converge, the computational time showed little gains.

Since the FPA algorithm solves a separable problem, we used a framework proposed in [21] to solve nonseparable optimization problems in the form of the portfolio optimization problem presented in our experiments. The results are similar to those presented in [19]. Finally, we performed the FPA and FPA2 with the sensor

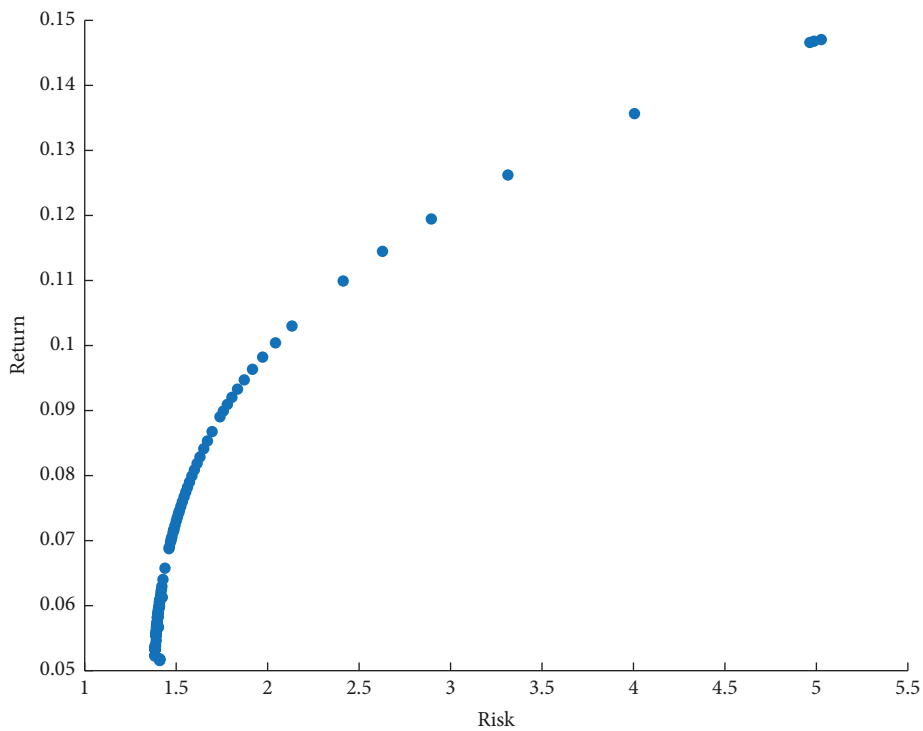

Figure 3: Ibovespa stocks.

Table 14: Results of the sensor placement problem.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
FPA2							
100	11.4	13	10	0.0	0.0	0.0	0
1000	16.5	18	15	0.1	0.1	0.0	0
10000	20.4	23	19	0.7	0.8	0.5	0
100000	24.3	25	23	7.2	9.5	6.6	0
1000000	28.8	30	28	87.9	103.2	83.6	0
Newton							
100	11.4	13	10	0.0	0.0	0.0	0
1000	16.5	18	15	0.1	0.1	0.1	0
10000	20.7	23	19	1.0	1.2	0.8	0
100000	24.3	25	23	10.3	13.3	9.5	0
1000000	28.8	30	28	123.2	192.3	109.7	0
Variable fixing							
100	11.4	13	10	0.0	0.0	0.0	0
1000	16.5	18	15	0.1	0.1	0.0	0
10000	20.7	22	19	0.8	1.0	0.6	0
100000	22.7	24	20	9.2	9.9	8.7	0
1000000	21.9	23	20	113.0	185.2	99.6	0
FPA							
100	11.4	13	10	0.0	0.0	0.0	0
1000	16.5	18	15	0.1	0.1	0.1	0
10000	20.4	23	19	1.0	1.1	0.8	0
100000	24.3	25	23	10.8	13.6	10.1	0
1000000	28.8	30	28	133.9	154.6	126.7	0
Secant							
100	37.2	61	28	0.0	0.0	0.0	0
1000	68.4	122	40	0.3	0.6	0.2	0
10000	442.7	1001	53	15.8	35.0	2.4	4

Table 14: Continued.

Dimension	Iterations			Time (sec)			Error
n	Avg	Max	Min	Avg	Max	Min	—
100000	1001.0	1001	1001	287.6	415.0	269.8	10
1000000	1001.0	1001	1001	3640.9	4487.0	3485.7	10
Median search							
100	8.0	8	8	0.0	0.0	0.0	0
1000	11.0	11	11	0.1	0.1	0.1	0
10000	14.0	14	14	1.0	1.1	0.8	0
100000	18.0	18	18	9.0	10.1	8.7	0
1000000	21.0	21	21	96.2	158.6	87.4	0

placement problem, and the proposed algorithm could solve all the generated problems faster than all compared algorithms.

Other acceleration techniques, as well as improvements in convergence analysis, are ongoing research.

Data Availability

The code and data used to support the findings of this study have been deposited in the GitHub repository and are available at https://github.com/jona04/scripts-fpa.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] K. M. Bretthauer and B. Shetty, "The nonlinear knapsack problem–algorithms and applications," *European Journal of Operational Research*, vol. 138, no. 3, pp. 459–472, 2002.

[2] R. Cominetti, W. F. Mascarenhas, and P. J. S. Silva, "A Newton's method for the continuous quadratic knapsack problem," *Mathematical Programming Computation*, vol. 6, no. 2, pp. 151–169, 2014.

[3] R. T. Münnich, E. W. Sachs, and M. Wagner, "Numerical solution of optimal allocation problems in stratified sampling under box constraints," *ASTA Advances in Statistical Analysis*, vol. 96, no. 3, pp. 435–450, 2012.

[4] M. Patriksson, "A survey on the continuous nonlinear resource allocation problem," *European Journal of Operational Research*, vol. 185, no. 1, pp. 1–46, 2008.

[5] M. Patriksson and C. Strömberg, "Algorithms for the continuous nonlinear resource allocation problem: new implementations and numerical studies," *European Journal of Operational Research*, vol. 243, no. 3, pp. 703–722, 2015.

[6] G. Kim and C. H. Wu, "A pegging algorithm for separable continuous nonlinear knapsack problems with box constraints," *Engineering Optimization*, vol. 44, no. 10, pp. 1245–1259, 2012.

[7] D. G. Anderson, "Iterative procedures for nonlinear integral Equations," *Journal of the ACM*, vol. 12, no. 4, pp. 547–560, 1965.

[8] A. C. Aitken, "XXV.—on Bernoulli's numerical solution of algebraic Equations," *Proceedings of the Royal Society of Edinburgh*, vol. 46, pp. 289–305, 1927.

[9] V. Aksenov, M. Chertov, and K. Sinkov, "Application of accelerated fixed-point algorithms to hydrodynamic well-fracture coupling," *Computers and Geotechnics*, vol. 129, Article ID 103783, 2021.

[10] S. Baumann and M. Klymak, "FixedPoint: a suite of acceleration algorithms with applications," 2021, https://cran.r-project.org/web/packages/FixedPoint/vignettes/FixedPoint.pdf.

[11] H. F. Walker and P. Ni, "Anderson acceleration for fixed-point iterations," *SIAM Journal on Numerical Analysis*, vol. 49, no. 4, pp. 1715–1735, 2011.

[12] K. C. Kiwiel, "Variable fixing algorithms for the continuous quadratic knapsack problem," *Journal of Optimization Theory and Applications*, vol. 136, no. 3, pp. 445–458, 2008a.

[13] A. Frangioni, C. Gentile, E. Grande, and A. Pacifici, "Projected perspective reformulations with applications in design problems," *Operations Research*, vol. 59, no. 5, pp. 1225–1232, 2011.

[14] Y. H. Dai and R. Fletcher, "New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds," *Mathematical Programming*, vol. 106, no. 3, pp. 403–421, 2006.

[15] P. M. Pardalos and N. Kovoor, "An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds," *Mathematical Programming*, vol. 46, no. 1-3, pp. 321–328, 1990.

[16] K. M. Bretthauer, B. Shetty, and S. Syam, "A branch- and bound-algorithm for integer quadratic knapsack problems," *ORSA Journal on Computing*, vol. 7, no. 1, pp. 109–116, 1995.

[17] K. C. Kiwiel, "Breakpoint searching algorithms for the continuous quadratic knapsack problem," *Mathematical Programming*, vol. 112, no. 2, pp. 473–491, 2007.

[18] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.

[19] E. M. Torrealba, J. G. Silva, L. C. Matioli, O. Kolossoski, and P. S. Santos, "Augmented Lagrangian algorithms for solving the continuous nonlinear resource allocation problem," *European Journal of Operational Research*, vol. 299, no. 1, pp. 46–59, 2022.

[20] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.

[21] J. Silva, A. Alves, P. Santos, and L. Matioli, "A new SVM solver applied to skin lesion classification," *Statistics, Optimization, and Information Computing*, In press.

[22] J. Snyman and A. Hay, "The spherical quadratic steepest descent (sqsd) method for unconstrained minimization with no explicit line searches," *Computers and Mathematics with Applications*, vol. 42, no. 1-2, pp. 169–178, 2001.

[23] A. Agnetis, E. Grande, and A. Pacifici, "Demand allocation with latency cost functions," *Mathematical Programming*, vol. 132, no. 1-2, pp. 277–294, 2012.