Scientific Research Publishing

American Journal of Operations Research, 2025, 15(6), 195-221
https://www.scirp.org/journal/ajor
ISSN Online: 2160-8849
ISSN Print: 2160-8830

A Perspective on Stochastic Search Efficiency via Quasigradient Techniques in Constrained Models

Gilberto Pérez-Lechuga

División de Estudios de Posgrado e Investigación, Instituto Tecnológico de Ciudad Madero, Tecnológico Nacional de México, Ciudad Madero, Tamaulipas, México
Email: perezlechuga59@gmail.com

How to cite this paper: Pérez-Lechuga, G. (2025) A Perspective on Stochastic Search Efficiency via Quasigradient Techniques in Constrained Models. *American Journal of Operations Research*, **15**, 195-221.
https://doi.org/10.4236/ajor.2025.156010

Received: August 30, 2025
Accepted: November 3, 2025
Published: November 7, 2025

Abstract

This article examines some of the properties of quasi-Fejer sequences when used in quasi-gradiental techniques as an alternative to stochastic search techniques for optimizing unconstrained mathematical programming models. The convergence and efficiency of the method are analyzed, and its potential use as an interior-point algorithm for optimizing integer linear programming models is explored, ensuring the feasibility of the solution at each stage of the search. To achieve this, it is proposed to remain within the feasible region by using small perturbations around the points found until convergence is reached. This alternative is compared with the traditional Branch and Bound method using software programs available for this purpose. The results obtained suggest that the technique, applied to models with few variables, is inefficient but is practical for large-scale models, since simple changes in the components of the located points generate a feasible sequence that almost always converges.

Keywords

Fejer Successions, Integer Programming, Stochastic Convergence Techniques, Random Search

1. Introduction

Generally, Random Search Techniques (RSM) are methods used for the optimization of some mathematical programming models using approximation sequences that improve in each iteration of the algorithm based on simple changes. Similarly, Stochastic Quasi-Gradient (SQG) methods are stochastic algorithmic procedures for solving general constrained optimization problems with non-differentiable and

non-convex functions. In deterministic models, this technique offers extensive possibilities for implementing alternative heuristics that seek to minimize classical computational complexity in models such as integer linear programming. This class of algorithms defines an exploration sequence similar to sequential adaptive learning and improves decisions based on data and simulations, which are known as Adaptive Monte Carlo Optimization [1].

The computational complexity of an integer linear programming model lies in the integrality restrictions imposed on the model, which force it to explore a discrete region with a number of combinations that grow exponentially as the model increases the number of variables. This means searching for solutions in a non-convex and non-continuous region, which greatly complicates the problem. Hence, they are classified as NP-hard problems, in which the time to convergence also grows exponentially. Methods for solving instances of integer programming models use exact algorithms and/or approximation methods that generally work using simple rules, almost always obtaining an approximate solution. In the first case, exact algorithms obtain exact solutions in reasonably finite times by reducing the search space, called cuts. Unfortunately, these methods work well for small problems, but are of no practical use since real problems require an enormous amount of time (exponential) to achieve convergence. Problems related to manufacturing, logistics, facility location, routing and more require millions of variables, almost always integers, to provide a satisfactory solution to the model [2]-[6]. Heuristics for integer programming are sometimes used with relative success because their approximation is not good, but they solve large-scale problems in reasonable times. Among the most popular are Local Branching, Relaxation-Induced Neighborhood Search and Variants.

An integer programming problem can be presented in several forms. In its purest form, all variables involved in the model must be integers; that is, it is a Pure Integer Linear Programming (*PILP*) model. In other cases, binary variables are required for yes-no decisions, in which case, the model is called Binary Linear Programming (*BLP*). Finally, a mixture of integer, continuous, and even binary variables in a single model produces a Mixed Integer Linear Programming (*MILP*) model. It has been intensively demonstrated that the computational complexity of an integer programming model (in any of its meanings) is classified as *NP* since the algorithm evolves in pseudo-polynomial time for any number of constraints [7].

Some approaches to the problem are based on the use of evolutionary algorithms that make use of selected nodes of a branching tree whose basic elements are the population, combination, mutation and selection [8] [9]. In some cases, integer optimization is used in novel areas such as bilevel optimization, where some variables are constrained to be the solution of another optimization problem [10]. Recently, learning techniques (supervised learning) have been introduced into this algorithm to improve its critical components [11]. These techniques are being considered as an alternative to solve combinatorial optimization (CO) problems and

for the moment, they represent a promising alternative idea to NP-complex problems. In this paper, we revisit a concept that has remained classic: the techniques developed by Ermoliev *et al.* [12] for optimization problems without the need for continuity or differentiability constraints, and orient them toward optimization models with integrity characteristics.

In the practice of engineering and science, it is frequently required to solve optimization models whose decision variables must be integers. This situation arises when non-fractional quantities are handled in decision-making models, such as the number of vehicles exported by a company, the number of people assigned to a manufacturing operation, or the number of meals to be served in a restaurant during a given period.

This paper analyzes and evaluates the alternative of using a perturbation and bounding technique at feasible points in an integer linear programming model. This allows the method to identify how a multivariate function changes in a specific direction, not just along the coordinate axes. Thus, using this rate of change, it is possible to find a direction of descent, allowing the procedure to remain within the search region while maintaining the feasibility of the solution.

Therefore, using a random search method from the first iteration, it is possible to introduce changes into the model at regular intervals driven by a known probability distribution.

Given the nature of the problem analyzed, the properties of quasi-Fejer sequences are useful to determine the convergence of the technique via the monotonicity properties of such sequences.

The problem is initially approached from the perspective of constructing a convex and bounded set from which discrete candidate points can be generated that can be evaluated to locate a descent direction. The ρ-dimensional sphere is ideal for this purpose. A hypersphere, or more commonly called an ρ-dimensional sphere, is a generalization of the circle (called a 2-sphere) and the usual sphere (called a 3-sphere) to dimensions $\rho \geq 4$. Therefore, the ρ-sphere is defined as the set of ρ-tuples of points $\left(x_1, x_2, \cdots, x_\rho\right)$ such that $x_1^2 + x_2^2 + \cdots + x_\rho^2 = R^2$. In optimization theory, an ρ-dimensional sphere takes on special importance because it defines a perfect convex and compact set.

For the above reasons, we take up the ideas developed in [13] [14] and extend them to the solution of cases of entire problems, we show the underlying theory in the proposal and we illustrate the results with a numerical example.

For this exploration, the following research questions will be answered:

1) Can an alternative metaheuristic be constructed that allows an approximation to the optimal value of an integer linear programming model using techniques?

2) What should be the shape of the descent direction?

3) How can a vector perturbation of them be achieved in order to find a sequence of integer points that satisfactorily converges to the desired optimum?

4) Would it be possible to apply the investigated method to large-scale models

commonly required in practical engineering and science applications while minimizing its inherent computational complexity?

To address this problem, the document has been organized as follows. In Section 2, the problem to be addressed is formally presented and its notation is defined. Section 3 illustrates how to apply the alternative studied and compares the results in simple visualization examples. A discussion of the findings is presented in Section 4; finally, Section 5 presents the conclusions of this work.

Below, we describe the sequence of steps used to solve the proposed instance.

2. Statement of the Problem

In mathematical programming, a heuristic is a technique used to approximate solutions to complex problems using simple rules that do not guarantee finding the optimal solution, but are good enough to achieve within a reasonable timeframe [15]. When a heuristic can be implemented as a computational algorithm, then it is called a metaheuristic [16].

In this part of the document, we are interested in analyzing an alternative method (heuristic) to optimize models of the following type

$$\text{Minimize } g(X) = c_1x_1 + c_2x_2 + \cdots + c_nx_n = \sum_{i=1}^{n} c_ix_i \tag{1}$$

where

- $g(X)$ is the objective function or utility function of the problem.
- c_i, $i = 1,2,\cdots,n$ define the cost coefficients of the utility function.
- $x_i \in \mathbb{Z}^+ \bigcup \{0\}$, $i = 1,2,\cdots,n$ are the decision variables (positive integers) of the problem

The restrictive set is defined by the matrix system given by

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \tag{2}$$

where a_{ij} are called the technological coefficients and b_i are the available resources. For simplicity, in the remainder of this document, the problem (*PILP*) will be represented in the following equivalent form

$$\text{Minimice } g(X) = \left\{ CX^t \mid AX = b, X \in \mathcal{D} \subset \mathbb{Z}^+ \bigcup \{0\} \right\}. \tag{3}$$

here, $\mathcal{D}$ is the feasible region where all the restrictions are met. A is a matrix of size $m \times n$, and b is a vector $m \times 1$. An overview of quasi-Fejer sequences is provided below.

A quasi-Fejer sequence is a sequence in a finite-dimensional Hilbert space that satisfies a Fejer monotonicity property, plus an additional error. Such sequences approach an accumulation point by dragging along an error that decreases with time. Formally, a sequence $\{X_k\}$ in a Hilbert space is Fejer monotone with respect to $\mathcal{D}$, if for each $X \in \mathcal{D}$ [17].

$$\|X_{k+1} - X_k\|^2 \leq \|X_k - X\|^2 + \epsilon_k$$

where $\sum \epsilon_k < \infty$.

Similarly, the concept of a quasi-gradient will be formally addressed as a statistical estimate of a true gradient. Thus, when searching within $\mathcal{D}$, an estimated approximation toward a downward direction in the first two search values will be used. The search is the updated based on the newly acquired information.

2.1. The Concept of Quasi-Gradient and Its Construction

Let $g(X)$ be a convex function not necessarily differentiable. The subgradient vector at the point $X=(x_1,\cdots,x_n)$ is any $\tilde{\nabla} g(X)$ vector that satisfies the inequality

$$g(Y)-g(X)\geq \left\langle \tilde{\nabla} g(X), Y-X \right\rangle \tag{4}$$

For any arbitrary, $Y\in\mathbb{R}^n$. The vector $\tilde{\nabla} g(X)$ forms a right angle with the normal to the supporting hyperplane of the set $\{Y \mid g(Y)<g(X)\}$, so if $g(X)$ is differentiable, then $\tilde{\nabla} g(X)$ coincides with the gradient of g in X, $\nabla g(X)$. Analogously, if $f(X)$ is convex, then g is a quasi-gradientin X if

$$g^{\mathrm{T}}(Y-X)\geq 0 \Rightarrow f(Y)\geq f(X)$$

Geometrically, g defines a supporting hyperplane to the sublevel set $\{x \mid f(X)\leq f(X_0)\}$. In this case, the set of quasigradients at X_0 forms a cone [18].

Random search techniques work from a sequence of random variables X defined on $\mathcal{D}$ that force it towards a limit point X^*. The randomness of the search consists of proposing an estimator of the subgradient vector that serves to obtain a direction of descent. Such estimator can be constructed using the Monte Carlo method, where the most important thing is to demonstrate that the proposed subgradient estimator is an expression of the type

$$\Xi(X)=\mathbf{E}\left[\nabla g(X)\right]=c_k\tilde{\nabla} g(X_k)+\Theta_k,\ k=0,1\cdots \tag{5}$$

where c_k is a non-negative number and Θ_k is a vector dimensionally compatible with the subgradient $\tilde{\nabla} g(x_k)$.

In this analysis, it is possible to use a variant of the technique to build at least an initial iteration to start the descent sequence, approaching the optimal point X^* step by step in such a way that in the k-th iteration, the point X_k is known and therefore, the next point X_{k+1} will be achieved through the classic approximation given by

$$X_{k+1}=X_k+\alpha_k\tilde{\nabla} g(X_k),\ \alpha>0,\ X_k\in\mathbb{Z}^+,\ k=1,2,\cdots \tag{6}$$

Because of the way in which these types of algorithms approach a solution, the criterion for stopping the search here is based on convergence in probability, that is,

$$\lim_{n\to\infty}\mathbf{P}\left[\left\|X_{k+1}-X_k\right\|\geq\epsilon\right]=0,\ \forall\epsilon>0. \tag{7}$$

This possibility is explored below.

2.2. The Mathematical Procedure

Consider a vector $\theta = (\theta_1, \cdots, \theta_n)$ whose components are independent and uniformly distributed random variables in $[-c_1, c_2]$. In this paper, a discrete uniform distribution is used to obtain a finite set of integer values uniformly distributed over the surface of the hypersphere.

Since there is no a priori information about the probability density that defines the search region, we will consider that the movement can occur in any direction with the same probability. Suppose that in iteration k, ρ_k samples of size s are available

$$\theta_{k_1}^{\mathrm{T}} = \left(\theta_{k_1}^1, \cdots, \theta_{k_1}^s\right)$$
$$\vdots$$
$$\theta_{k_i}^{\mathrm{T}} = \left(\theta_{k_i}^1, \cdots, \theta_{k_i}^s\right)$$
$$\theta_{k_{\rho_k}}^{\mathrm{T}} = \left(\theta_{k_{\rho_k}}^1, \cdots, \theta_{k_{\rho_k}}^s\right)$$

If X_k is a vector given at iteration k with $\Delta_k > 0$, then an estimator $\xi(X_k)$ of the subgradient of g in X_k can be written as

$$\begin{aligned}\xi(X_k) &= \sum_{i=1}^{\rho_k} \frac{g\left(X_k + \Delta_k \theta_{k_i}^{\mathrm{T}}\right) - g(X_k)}{\Delta_k} \theta_{k_i} \\ &= \frac{g\left(X_k + \Delta_k \theta_{k_1}^{\mathrm{T}}\right) - g(X_k)}{\Delta_k} \theta_{k_1} + \cdots + \frac{g\left(X_k + \Delta_k \theta_{k_{\rho_k}}^{\mathrm{T}}\right) - g(X_k)}{\Delta_k} \theta_{k_{\rho_k}},\end{aligned} \tag{8}$$

thus, assuming convexity in g, we have that

$$\frac{g\left(X_k + \Delta_k \theta_{k_i}^{\mathrm{T}} - g(X_k)\right)}{\Delta_k} \theta_{k_i} \geq \left\langle \tilde{\nabla} g(X_k), \theta_{k_i}^{\mathrm{T}} \right\rangle \theta_{k_i}.$$

So, applying the mathematical expectation operator, we have to

$$\begin{aligned}\mathbf{E}\left\{\frac{g\left(X_k + \Delta_k \theta_{k_i}^{\mathrm{T}} - g(X_k)\right)}{\Delta_k} \theta_{k_i}\right\} &\geq \mathbf{E}\left\{\left\langle \tilde{\nabla} g(X_k), \theta_{k_i}^{\mathrm{T}} \right\rangle \theta_{k_i} \mid X_k\right\} \\ &= \mathbf{E}\left\{\left\langle \theta_{k_i}^{\mathrm{T}}, \theta_{k_i} \right\rangle \tilde{\nabla} g(X_k) \mid X_k\right\} \\ &= \frac{n}{12}(c_2 - c_1)^2 \tilde{\nabla} g(X_k)\end{aligned}$$

Therefore,

$$\mathbf{E}\left\{\xi(X_k) \mid X_k\right\} = \frac{n}{12}(c_2 - c_1)^2 \tilde{\nabla} g(X_k) + W_k \tag{9}$$

From the above, it follows that if, $Y = \left(\theta_{k_i}^j\right)^2$ $j = 1, \cdots, n$, then Y has a density given by

$$f_Y(y) = \begin{cases} \dfrac{1}{(c_2 - c_1)\sqrt{y}}, & \text{if } 0 \leq y \leq \dfrac{1}{4} \\ 0, & \text{otherwise} \end{cases}$$

and from the previous assumption, it follows that W_k is uniformly bounded, *i.e.*, $\|W_k\| \le \epsilon$, $\epsilon > 0$.

To facilitate the analysis process, we now simplify the search space and define the projection operator. Let us define the set as closed and convex $\mathcal{D} = \{X \mid a \le X \le b\}$. Let $\pi(X)$ be the projection operator on $\mathcal{D}$; that is, for any $X \in \mathbb{R}^n$, $\pi(X) \in \mathcal{D}$ and

$$\|X - \pi_{\mathcal{D}}(X)\| = \min_{Y \in \mathcal{D}} \|X - Y\|$$

Let the random sequence of points X_k be defined as

$$X_{k+1} = \Pi_{\mathcal{D}}(X_k - \alpha_k \gamma_k \xi_k), \ k = 1, \cdots, \tag{10}$$

where X_0 is an arbitrary point for which $\mathbf{E}\{\|X_0\|^2\} = \text{cte} < \infty$, α_k is the step length, γ_k is a normalization factor and $\xi_k = (\xi_{k_1}, \cdots, \xi_{k_n})$ is a random vector whose conditional mathematical expectation is given by

$$\mathbf{E}\{\xi_k \mid X_0, \cdots, X_k\} = c_k \tilde{\nabla} g(X_k) + \Theta_k, \ k = 1, \cdots, \tag{11}$$

here, c_k is a non-negative number, $\Theta_k = (\theta_{k_1}, \cdots, \theta_{k_n})$ is a vector, $\tilde{\nabla} g(X)$ is a subgradient, that is, the vector ξ_k satisfies a relation of the form

$$\mathbf{E}\{\xi(X) \mid H(X)\} = c \tilde{\nabla} g(X) + \Theta. \tag{12}$$

Notice that, when $\mathcal{D} = \mathbb{R}^n$ and $\pi(X) = X$, Equation (10) can be used to optimize models of the type (1) and the method is called the *generalized stochastic quasi-gradient method*. The results presented below are based on the iconic work of Ermoliev [19].

Lemma 1 (Convergence). Suppose that the values of h_k are known such that $\mathbf{E}\{\|\xi_k\|^2 \mid X_0, \cdots, X_k\} \le h_k^2 \le M_B < \infty$, $i = 1, \cdots, k$, and also $\|X_k\| \le B < \infty$, $i = 1, \cdots, k$. Let be the normalization factor γ_k that satisfies the equation

$$0 \le \gamma_k (\tau_k \|X_k\| + h_k) < \infty,$$

where $h_k(X_0, \cdots, X_k)$, $\tau_k = 1$ if $\|\Theta_k\| > 0$, and $\tau_k = 0$, if $\|\Theta_k\| = 0$. Let the quantities

$$\alpha_k \ge 0, \ c_k \ge 0, \ \sum_{k=0}^{\infty} \alpha_k r_k < \infty, \ \sum_{k=0}^{\infty} \alpha_k^2 r_k < \infty, \tag{13}$$

then, the sequence of points defined by Equation (10) is a quasi-Fejer sequence with respect to the set $\mathcal{D}$. Even more, if it satisfied

$$\sum_{k=0}^{\infty} \alpha_k l_k = \infty \tag{14}$$

then, the sequence $\{X_k\}$ converges globally to the solution of the problem

$$\text{Minimize}_{\mathbf{X}} \ \mathbf{E}_W \{\psi(X, w) \mid X \in \mathcal{D}\}. \tag{15}$$

Proof. Let X^* be an arbitrary solution to the problem (15), we have to

$$\begin{aligned} \|X^* - X_{k+1}\|^2 &= \|X^* - \Pi_D(X_k - \alpha_k \gamma_k \xi_k)\|^2 \\ &\le \|X^* - X_k + \alpha_k \gamma_k \xi_k\|^2 \\ &= \|X^* - X_k\|^2 + 2\alpha_k \gamma_k \langle \xi_k, X^* - X_k \rangle + \alpha_k^2 \gamma_k^2 \|\xi_k\|^2. \end{aligned}$$

Taking the mathematical expectation on both sides of the equality, we have

$$\begin{aligned}&\mathbf{E}\left\{\left\|X^*-X_{k+1}\right\|^2 \mid X_0,\cdots,X_k\right\}\\&\le\left\|X^*-X_k\right\|^2+2\alpha_k\gamma_k c_k\left\langle\tilde{\nabla}g\left(X_k\right),X^*-X_k\right\rangle+2\alpha_k\gamma_k\left\langle\Theta_k,X^*-X_k\right\rangle\\&\quad+\alpha_k^2\gamma_k^2\mathbf{E}\left\{\left\|\xi_k\right\|^2 \mid X_0,\cdots,X_k\right\},\end{aligned}$$

where

$$\begin{aligned}\mathbf{E}\left\{\left\|\bar{X}\right\|\right\}&=\mathbf{E}\left\{\left\|\left(\bar{X}_1(\omega),\cdots,\bar{X}_n(\omega)\right)^{\mathrm{T}}\right\|\right\}\\&=\int_\Omega\sqrt{\bar{X}_1^2(\omega),\cdots,\bar{X}_n^2(\omega)}\,\mathbf{P}(\mathrm{d}\omega).\end{aligned}$$

here, Ω is sample space corresponding to the probability space (Ω,F,P). Applying the Cauchy-Schwarz inequality and considering that $g\left(X^*-g\left(X_k\right)\right)\le 0$, we have to

$$\begin{aligned}&\mathbf{E}\left\{\left\|X^*-X_{k+1}\right\|^2 \mid X_0,\cdots,X_k\right\}\\&\le\left\|X^*-X_k\right\|^2+2\alpha_k\gamma_k c_k\left[g\left(X^*\right)-g\left(X_k\right)\right]+2\alpha_k\gamma_k\left\|\Theta_k\right\|\left\|X^*-X_k\right\|\\&\quad+\alpha^2\gamma^2\mathbf{E}\left\{\left\|\xi_k\right\|^2 \mid X_0,\cdots,X_k\right\}\\&\le\left\|X^*-X_k\right\|^2+2\alpha_k\gamma_k\left\|\Theta_k\right\|\left[\left\|X^*\right\|+\left\|X^*\right\|\right]+\alpha^2\gamma^2\mathbf{E}\left\{\left\|\xi_k\right\|^2 \mid X_0,\cdots,X_k\right\}\\&\le\left\|X^*-X_k\right\|^2+2\alpha_k\gamma_k\left[\gamma^*\left\|X^*\right\|+\gamma_k\left\|X_k\right\|\right]+\alpha_k^2\gamma_k^2M_B\\&\le\left\|X^*-X_k\right\|^2+2\alpha_k\gamma_k\left[\gamma^*\left\|X^*\right\|+\gamma^*B\right]+\alpha^2\gamma^*M_B.\end{aligned}$$

The inequalities found and the conditions defined in Equation (13) prove the first part of the theorem. Now, it will be proven that if the conditions of Equation (14) are met, then one of the limit points of the succession $\left\{X_s(\omega)\right\}$, for almost all, ω belongs to the set of solutions to the problem (1). Applying mathematical expectation again, we have to

$$\begin{aligned}\mathbf{E}\left\{\left\|X^*-X_{k+1}\right\|^2\right\}\le\left\|X^*-X_k\right\|^2+2\textstyle\sum_{k=0}^{s}\alpha_k l_k\mathbf{E}\left\{\gamma_k\left\langle\tilde{\nabla}g\left(X_k\right),X^*-X_k\right\rangle\right\}\\+2\left[\gamma_k^*\left\|X^*\right\|+\gamma^*B\right]\textstyle\sum_{k=0}^{s}\alpha_k r_k+\sum_{k=0}^{s}\alpha_k^2\gamma^*M_B\end{aligned}\tag{16}$$

From (16), it follows that $\mathbf{E}\left\{\left\|X^*-X_{k+1}\right\|^2\right\}$ is uniformly bounded and

$$\sum_{k=0}^{\infty}\alpha_k l_k\mathbf{E}\left\{\gamma_k,\left\langle\tilde{\nabla}g\left(X_k\right),X^*-X_k\right\rangle\right\}\ge-\infty.$$

Since that $\sum_{k=0}^{\infty}\alpha_k l_k=\infty$, we have to $\mathbf{E}\left\{\gamma_k\left\langle\tilde{\nabla}g\left(X_k\right),X^*-X_k\right\rangle\right\}\to 0$ when $k\to\infty$. Note also that there exists a subsequence $\left\{s_t\right\},t=0,1,\cdots$ for which

$$\gamma_{st}(\omega)\left\langle\tilde{\nabla}g\left(X_{st}(\omega)\right),X^*-X_{st}(\omega)\right\rangle\to 0.$$

with probability one, according to $t\to 1$. It is concluded then that, for almost all ω, the sequence $\left\{X_k(\omega)\right\}$ is bounded, that is, for almost all ω $\left\langle\tilde{\nabla}g\left(X_{k_t}(\omega)\right),X-X_{k_t}(\omega)\right\rangle\to 0$. Then, as $t\to\infty$, the sequence $X_{k_t}(\omega)$ converges to the solution of the problem (15), the theorem is proven. □

2.3. Generating a Descent Trajectory

A first approach to what could be a search method using the technique is to use a variant of Equation (11) by constructing a dome around the point X_k; that is, create a hypersphere of dimension ρ centered on it and generate a sample of points uniformly distributed on the surface of the hypersphere of unit radius and use the distance between the center and the surface of the hypersphere as the difference between X_k and X_{k+1} to obtain an approximation to Equation (6). The formal ideas are discussed below.

Let's consider a hypersphere of dimension ρ in which we will obtain a number of points uniformly distributed over its surface (**Figure 1**).

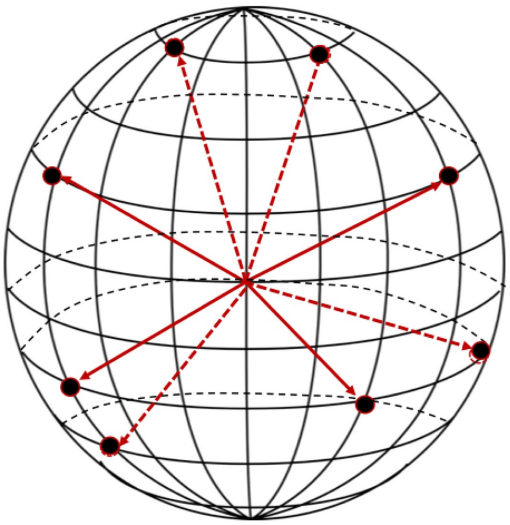

Figure 1. ρ discrete points uniformly distributed over the surface of a hypersphere in $\mathbb{R}^{\rho}$.

Let $X=\left(X_1,X_2,\cdots,X_{\rho}\right)$ be a random vector where $X_1,X_2,\cdots,X_{\rho}$ are independent and continuous random variables defined over a probability space $\left(\Omega,\mathfrak{F},S\right)$ with joint density function given by

$$f_{X_1,X_2,\cdots,X_{\rho}}\left(x_1,x_2,\cdots,x_{\rho}\right)=\prod_{i=1}^{\rho}f_i\left(x_i\right), \tag{17}$$

By the continuity hypothesis of X, $F_{X_1,X_2,\cdots,X_{\rho}}\left(x_1,x_2,\cdots,x_{\rho}\right)$ is a non-decreasing function and therefore, for each X_i there is the inverse function $\xi_i=F_{X_i}^{-1}\left(U\right)$ defined for any value of $U\in\mathcal{U}\sim\left[0,1\right]$ such that

$$\xi_i=F_{X_i}^{-1}\left(U\right)=\inf\left\{x:F_{X_i}\left(x\right)\geq U\right\}, \tag{18}$$

The generation of a random vector $Y\in\mathbb{R}^{\rho}$ on the surface of a unit hypersphere in the same dimension is given by **Algorithm 1** shown below [20].

Thus, given an initial value X_k (which is the center of the unitary hypersphere), generate ρ points uniformly distributed on its surface and apply the following criterion to select the consecutive value (applicable to the minimization case).

Algorithm 1. Algorithm for generating vectors on an ρ-dimensional unit hypersphere.

Require: Generate $U_1, \ldots, U_\rho$ from $\mathcal{U}(0,1)$
Ensure: $Z \in \mathbb{R}^\rho$
while $Y^2 < 1$ **do**
$X_i \leftarrow 1 - 2U_i, \quad i = 1, \ldots, \rho$
$Y^2 \leftarrow \sum_{i=1}^{n} X_i^2$
$Z_i \leftarrow \frac{X_i}{Y}$
$Z = (Z_1, \ldots, Z_\rho)$
if $Y^2 \geq 1$ **then** Discard the values of U_i and obtain a new sample of size ρ. Go back to the beginning of the algorithm.
end if
Update the Z parameter values
end while

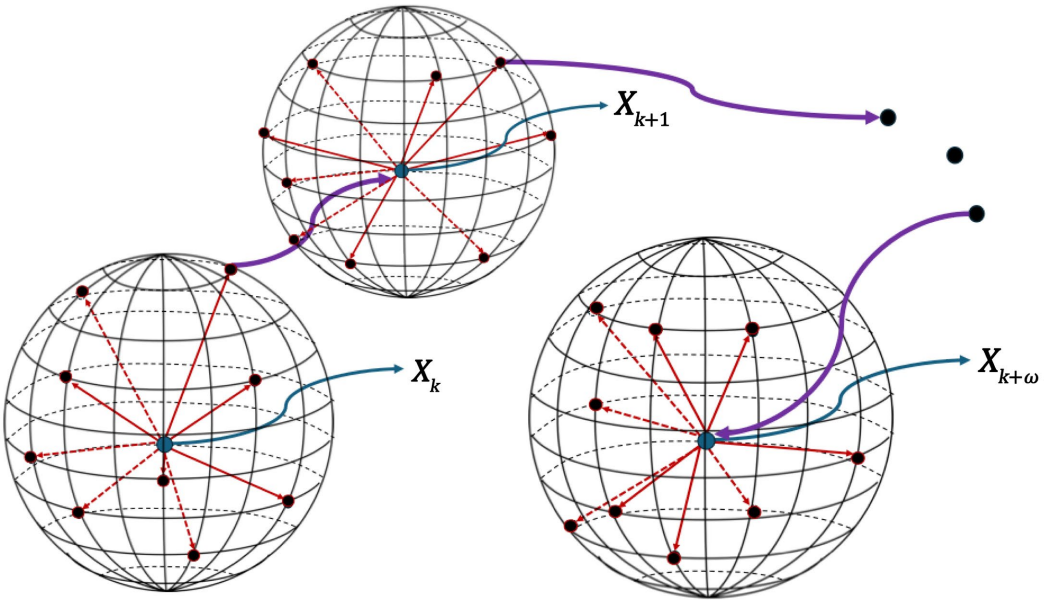

Figure 2. Trajectory search method using a unit hypersphere.

This is, the new value X_{k+1} becomes the center of the new hypersphere and the process is repeated (**Figure 2**). The following sequencing process is then generated

$$X_{k+1} \leftarrow X_k \text{ if } \left[g\left(X_{k+1}\right) < g\left(X_k\right) \right], \quad k = 1, \cdots, \tag{19}$$

Otherwise, the value of X_k is retained.

It can be shown that the acceptance-rejection method to generate the corresponding lotteries is highly inefficient. Its effectiveness is given by

$$\eta(\rho) = \frac{\text{Volume of the hypersphere}}{\text{Volume of the hypercube}} = \frac{1}{\rho 2^{\rho-1}} \frac{\pi^{\rho/2}}{\Gamma\left(\rho/2\right)}$$

where $\Gamma(\cdot)$ is the gamma function. It is easy to verify that for $\rho > 4$, the algorithm becomes inefficient and therefore impractical for large-scale problems (**Table 1**). For this reason, it is of practical use to locate at most $\rho = 4$ points uniformly distributed on the surface of the hypersphere.

Table 1. Efficiency of the algorithm as a function of the number of points required.

ρ	1	2	3	4	5	6	7	8
$\eta(\rho)$	1.0000	0.7854	0.5236	0.3084	0.1645	0.0807	0.0369	0.0159

Thus, it will be perturbing just some of the components x_k of X_k in the following manner

$$\hat{x}_k = x_k \pm \beta_k, \text{ for some } x_k \in X_k, \ \beta_k \sim \lfloor \mathcal{U}[-1,2] \rfloor, \ k = 0,1,\cdots \tag{20}$$

where $\lfloor r \rfloor$ means the largest integer less than or equal to r, and the draw of the random variable β_k provides the lotteries of the integer values −1, 0, 1, expanding the search in the neighborhood given by Equation (20). Therefore, we now have the perturbed sequence given by

$$\hat{X}_{k+1} = \left\lfloor \hat{X}_k - \alpha_k \tilde{\nabla} g\left(\hat{X}_k\right) \right\rfloor, \ k = 0,1,\cdots \tag{21}$$

with $X_0 \in \mathcal{D}$ known, $\hat{X}_k$ is the new perturbed vector containing one or more perturbed components $\hat{x}_k$. This means that the search should be focused on the direction where $g(X_k)$ changes value as quickly as possible. The guideline for selecting the appropriate component is to perturb the value of x_k satisfying the requirements of Equation (19) and using $\hat{X}_k$ instead of X_k.

Algorithm 2. Pseudocode associated with the proposal.

Require: $X_a, X_b \in \mathcal{D}, \ \epsilon = 0.01$ ▷ Start of the algorithm
Ensure: $\hat{X}_{k+1}$,
  $k \leftarrow 0$
  $\gamma_k \leftarrow \| X_b - X_a \|^2$
  $\tilde{\nabla} g(X_k) \leftarrow \frac{g(X_b) - g(X_a)}{\gamma_k}$
  $\lfloor X_k \rfloor \leftarrow X_a$ if $\min[g(X_a), g(X_b)] = g(X_a)$; otherwise $\lfloor X_k \rfloor \leftarrow X_b$
  $\alpha_k \leftarrow \frac{\gamma_k}{\|X_k\|^2}$
  $X_{k+1} \leftarrow \lfloor X_k - \alpha_k \tilde{\nabla} g(X_k) \rfloor$
  if $\lfloor X_{k+1} \rfloor \in \mathcal{D}$ **then**
    $g(\cdot) \leftarrow \lfloor X_k \rfloor, \quad g(\cdot) \leftarrow \lfloor X_{k+1} \rfloor$
    while $\epsilon_k \leq \epsilon$ **do**
      $\epsilon_k \leftarrow | \, g(\lfloor X_{k+1} \rfloor) - g(\lfloor X_k \rfloor) \, |$
      $\beta_k \leftarrow (3U - 1), \ U \sim \mathcal{U}(0,1)$
      $k = k + 1$
      $\hat{X}_k \leftarrow \lfloor \beta_k \times x_s \rfloor$, for some $x_s \in X_k$ such that $g(\lfloor \hat{X}_k \rfloor) < g(\lfloor X_k \rfloor)$, $\hat{X}_k \in \mathcal{D}$
      $\lfloor X_{k+1} \rfloor \leftarrow \lfloor X_k \rfloor$
      $\lfloor X_k \rfloor \leftarrow \lfloor X_{k-1} \rfloor$
    end while
  else if $\lfloor \hat{X}_{k+1} \rfloor \notin \mathcal{D}$ **then** ▷ Return to the beginning of the algorithm
  end if

In this, α_k satisfies the following conditions:

$$\alpha_k = \frac{\gamma_k}{\left\| g\left(X_k\right) \right\|^2}, \text{ or equivalently } \left\| X_{k+1} - X_k \right\|^2 = \gamma_k \tag{22}$$

$$\alpha_k \geq 0, \ \sum_{k=1}^{\infty} \alpha_k^2 < \infty, \ \sum_{k=1}^{\infty} \alpha_k = \infty. \tag{23}$$

and the quasi-gradient estimator is given by

$$\tilde{\nabla} g\left(X_k\right) = \frac{g\left(\hat{X}_{k+1}\right) - g\left(\hat{X}_k\right)}{\left\| \left(\hat{X}_{k+1}\right) - \left(\hat{X}_k\right) \right\|^2} \tag{24}$$

Equations (22) and (23) show the conditions that must be imposed on the components of the estimator in order to achieve their concurrence at a minimum value. Such values are necessary in the development of a Fejer sequence to guarantee the monotonicity of its convergence and have been widely demonstrated in [21] and Theorem 1 of this document.

Algorithm 2 and the pseudocode shown in **Figure 3** illustrate the steps followed in this process.

Once this outline is complete, we now proceed to test the proposal in the next section.

3. Numerical Results

To illustrate the use of the algorithm, the corresponding pseudocode and the algorithm associated with the proposal are shown below in **Figure 3**.

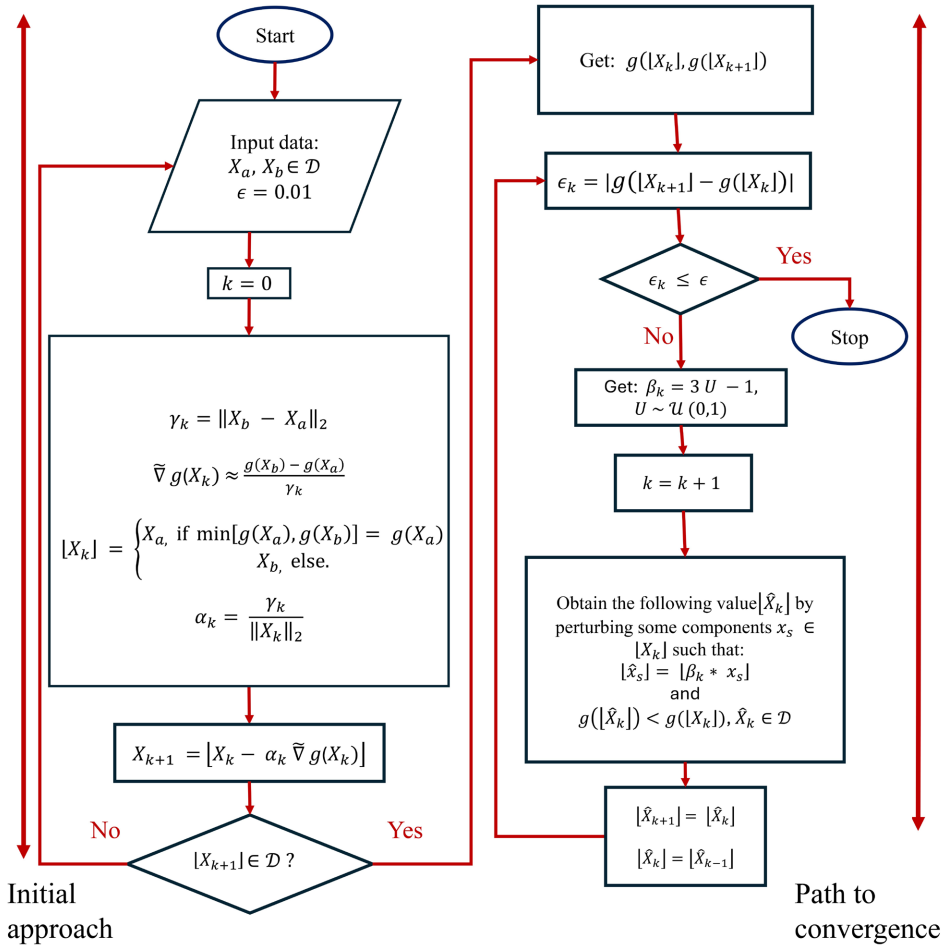

Figure 3. Pseudocode associated with the proposal.

The use of the proposal is illustrated below with a numerical example.

- **Model 1**

$$\begin{aligned}\text{Minimize } g(X) = {} & 2x_1 - x_2 - 6x_3 + 8x_4 + 15x_5 + 17x_6 - 21x_7 + 14x_8 + 16x_9 \\ & + 6x_{10} - 6x_{11} + 2x_{12} + 7x_{13} + 16x_{14} + 11x_{15} - 5x_{16} - 3x_{17} \\ & - 2x_{18} - 9x_{19} + 16x_{20} + 11x_{21} + 21x_{22} + 14x_{23} - 5x_{24} + 5x_{25} \\ & + 4x_{26} + 2x_{27} + 8x_{28} - 7x_{29} + 3x_{30}\end{aligned}$$

Subject to:

$$\begin{aligned}& 12x_2 + 6x_3 - 8x_4 + x_8 + 11x_9 + 4x_{10} + 7x_{11} - 4x_{12} + x_{13} + 5x_{15} \\ & - 4x_{17} + 2x_{21} + x_{22} - 2x_{23} + 4x_{24} + 6x_{26} - x_{28} + 8x_{29} + 3x_{30} \geq 750 \\ & x_1 - x_2 + 4x_3 + 7x_4 + 2x_5 + x_6 - 8x_7 + 11x_{14} + 9x_{16} + 11x_{18} \\ & + 21x_{22} - x_{25} + 8x_{27} + x_{29} \leq 1200 \\ & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} \\ & + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} - x_{21} - 4x_{22} \\ & - 9x_{23} - 11x_{24} - x_{25} - x_{26} - 2x_{27} - 8x_{28} + x_{29} + 2x_{30} \leq 0 \\ & x_2 + x_5 + x_8 + x_{11} + x_{14} + x_{17} + x_{20} - x_{23} - x_{26} - x_{29} \geq -1 \\ & 4x_4 + 8x_8 + 12x_{12} + 16x_{16} + 20x_{20} + 24x_{24} + 28x_{28} - 30x_{30} \leq 1350 \\ & x_1 \leq 4; x_2 \leq 6; x_3 \leq 8; x_4 \leq 10; x_5 \leq 14; x_6 \leq 18; x_7 \leq 22; x_8 \leq 26 \\ & x_9 \leq 20; x_{10} \leq 12; x_{11} \leq 16; x_{13} \leq 18; x_{14} \leq 16; x_{15} \leq 18; x_{16} \leq 20 \\ & x_{17} \leq 22; x_{18} \leq 26; x_{19} \leq 24; x_{20} \leq 24 \\ & x_{ij} \in \mathbb{Z}^+ \ \forall i, j\end{aligned}$$

The exact solution to this instance is shown in Table 2. This is reached after 63 iterations using a standard scientific method of Integer Linear Programming (*ILP*) via LINGO [22], with $g(X^*) = -1816$.

Table 2. Exact solution of the proposed instance.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
2	6	8	0	0	0	22	0	0	0
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
16	0	0	0	0	20	22	26	24	0
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{25}	x_{27}	x_{28}	x_{29}	x_{30}
0	0	0	136	0	0	0	0	45	75

To illustrate the use of this algorithm, the initial steps of the algorithm applied to the previous example are indicated below (see Table A1: Initial solutions).

1) Let X_A and $X_B \in D$ be defined as follows, with $g(X_A) = 2069$ and $g(X_B) = 2022$. Thus, for $\epsilon = 0.01$ and $k = 0$, we have

$$\begin{aligned}X_A = {} & 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 \\ & 16,17,18,19,20,21,22,23,24,25,26,27,28,29,30 \\ X_B = {} & 1,2,3,4,5,6,6,7,8,10,11,11,13,14,15 \\ & 15,17,17,18,19,21,22,22,24,24,25,26,27,29,29\end{aligned}$$

2) From the above, it is verified that $\gamma_k = \|X_b - X_a\|_2 = 3.7417$.

3) Similarly, $\tilde{\nabla}(X_0) \approx \dfrac{2069-2022}{3.7417} = 12.5611$.

4) It is also verified that $\min\{g(X_A), g(X_B)\} = 2069$, therefore $X_1 = X_A$, and

$$\alpha_k = \frac{\gamma_k}{\|X_1\|_2} = \frac{3.7417}{97.2368} = 0.0384$$

5) Therefore, for $k=2$, we obtain $X_2 \in \mathcal{D}$ given by

$$X_2 = \lfloor X_1 - 0.0384(12.5611) \rfloor, \text{ with } g(X_2) = 1936$$

6) The incremental value obtained is $\epsilon = \|X_2 - X_1\|_2 = 4$. Because $\epsilon > 0.01$, the specified the process continues.

7) What if $X_k \notin \mathcal{D}$? Proceed to the bounding phase as follows: get the random variable $\beta_k = 3U - 1$ such that $U \sim \mathcal{U}(0,1)$ and modify X_k as shown below

$$\hat{X}_k = \lfloor \beta_k * X_k \rfloor$$

until $\hat{X}_k \in \mathcal{D}$. Replace $\hat{X}_k$ by X_{k+1} and continue with the process.

The quantity $(3U-1)$ allows us to recognize the neighborhood around X_k and to carefully advance in the region, avoiding falling into points outside the boundary of $\mathcal{D}$. Although slow, this procedure allows a safe advance towards an approximate convergence. When the sequence approaches the boundary of $\mathcal{D}$, it is highly probable to generate sequences of infeasible points, having to further reduce the size of the search. Finally, an experimental strategy found suggests perturbing only some components of X_k as a directional derivative. The way of choosing the components to be perturbed obeys the criterion $\hat{x}_s = \lfloor \beta_k * x_s \rfloor$, such that $g\left(\lfloor \hat{X}_k \rfloor\right) < g\left(\lfloor X_k \rfloor\right)$.

Clearly, if the sequence $\{\alpha_k\}_k$ is such that $\{\alpha_k\}_k < 1$, then it satisfies the conditions imposed in Equations (22) and (23).

The rest of the solutions and the convergence to the optimal solution are shown in Table A1 of **Appendix**.

The search behavior and its convergence are shown in Figure 4. It shows how quickly the algorithm progresses in its first attempts to locate solutions better than the original. However, as the algorithm progresses, the search slows, and convergence encounters increasing difficulties in locating a new transfer point. This is because both the feasibility and convergence conditions must be met. In particular, the algorithm slows down when approaching points located on or near the boundary of $\mathcal{D}$, since neighborhoods are often located in infeasible zones. However, once the subsequent points enter $\mathcal{D}$, the algorithm appears to advance more quickly. This also explains why it is slow when approaching the optimal value, since, as is known, it lies in a corner of the simplex formed by its constraints.

Using the described technique, the convergence of the method required 116 iterations. Figure 5 shows the graph of the norm $\left\|\hat{X}_{k+1} - \hat{X}_k\right\|^2$ recorded during the iterations of the algorithm.

Finally, Figure 6 shows the behavior of the $\alpha(k)$ value throughout the evolution of the search.

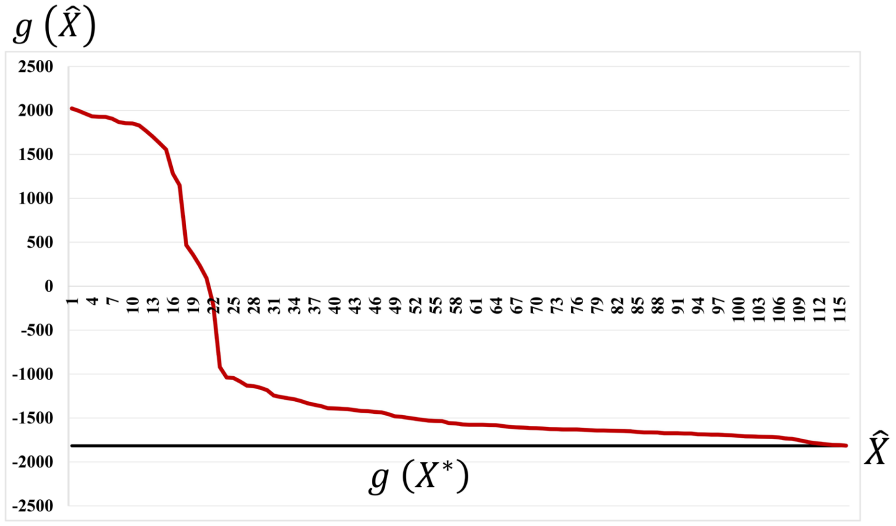

Figure 4. Required iterations and speed of convergence of the objective function.

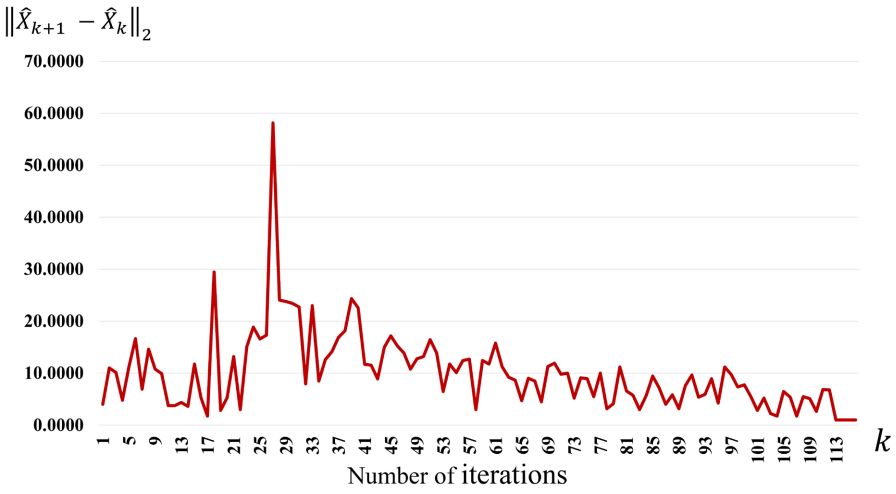

Figure 5. Difference in norms during the convergence process.

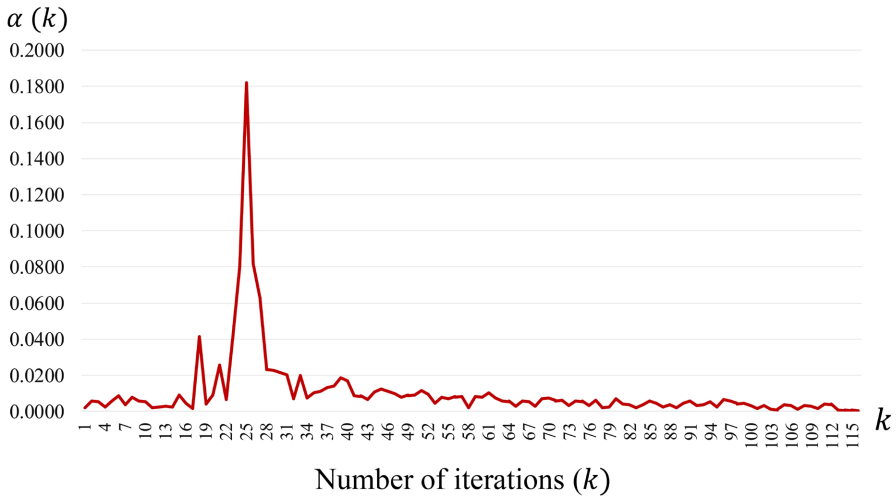

Figure 6. Values of the parameter α_k as a function of the number of iterations.

4. Discussion and Statistical Analysis of the Results

The results found suggest that the proposed method is feasible and constitutes an alternative for large-scale integer models. However, there is still considerable work to be done in the form of selection and creation of the $\hat{x}_k$ sequences since a direct comparison between $g\left(\hat{X}_{k+1}\right)-g\left(\hat{X}_k\right)$ is inefficient. So far, ensuring that $\hat{X}_k \in \mathcal{D}$ is only achieved through a feasibility test. Therefore, another line of research in this regard consists of developing a method in which the $\left\{\hat{X}_k\right\}$ sequence remains feasible at all times.

A simple way to evaluate the efficiency of the method is to determine the mathematical expectation of the differences in increments in a single step $\Delta_k = g\left(X_{k+1}\right)-g\left(X_k\right)$ and obtain the estimator

$$\eta_k = -\frac{\mathbf{E}\{\Delta_k\}}{\mathbf{E}\{N_k\}}, \tag{25}$$

where N_K is the number of points that must be evaluated to locate a descent direction at iteration k. The approach of the research would be oriented to maximize the value of η_k.

An approximation to the efficiency function is as follows. Note that $X_{k+1} = X_k + \Delta X_k$, therefore

$$g\left(X_{k+1}\right) = g\left(X_k\right) + \Delta X_k$$

Then, by the convexity of g, we have that

$$\begin{aligned} g\left(X_{k+1}\right) &= g\left(X_k\right) + \Delta X_k \\ &= g\left(X_k\right) + \langle \Delta X_k, \tilde{\nabla} g\left(X_k\right)\rangle + \delta\left(\Delta X_k\right) \\ &= g\left(X_k\right) + \left\|\Delta X_k\right\|\left\|\tilde{\nabla} g\left(X_k\right)\right\|\cos(\varphi) + \delta\left(\Delta X_k\right) \end{aligned} \tag{26}$$

where $\cos(\varphi)$ is the cosine of the angle formed by the unit sphere by the vectors ΔX_k and $\tilde{\nabla} g\left(X_k\right)$. These vectors are unitary and start from the center of the sphere, generating points on its surface. Similarly, the function $\delta\left(\Delta X_k\right)$ is such that $\delta\left(\Delta X_k\right) \to 0$ when $\left(\Delta X_k\right) \to 0$.

Then

$$\Delta X_k = X_{k+1} - X_k = -\alpha \tilde{\nabla} g\left(X_k\right)$$

Thus, by conveniently substituting in Equation (26), we have that

$$g\left(X_{k+1}\right) = g\left(X_k\right) + \alpha\left\|\tilde{\nabla} g\left(X_k\right)\right\|_2 \cos(\varphi) - \delta\left(\alpha \tilde{\nabla} g\left(X_k\right)\right).$$

where for a small ΔX_k, we have that

$$\Delta g_k = g\left(X_{k+1}\right) - g\left(X_k\right) = \alpha\left\|\tilde{\nabla} g\left(X_k\right)\right\|_2 \cos(\varphi).$$

Thus, by Equation (25), it is concluded that

$$\begin{aligned} \eta &= -\frac{\mathbf{E}\{\Delta g_k\}}{\mathbf{E}\{N_k\}} = -\frac{\mathbf{E}\left\{\alpha\left\|\tilde{\nabla} g\left(X_k\right)\right\|_2 \cos(\varphi)\right\}}{\mathbf{E}\{m\}} \\ &= -\frac{\alpha}{m}\left\|\tilde{\nabla} g\left(X_k\right)\right\|_2 \mathbf{E}\left[\cos(\varphi)\right], \end{aligned} \tag{27}$$

where m is the number of points evaluated before finding a descent direction, and $-\pi/2 \le \varphi \le \pi/2$. It can be shown that the density of the random angle φ for an n-dimensional hypersphere is given by [23].

$$\zeta_n(\varphi) = \frac{\sin^{n-2}(\varphi)}{\int_0^{\pi} \sin^{n-2}(\varphi)\,\mathrm{d}\varphi} = B_n \sin^{n-2}(\varphi)$$

where

$$B_n = \frac{\Gamma(n/2)}{\sqrt{\pi}\,\Gamma\left[(n+1)/2\right]},$$

and $\Gamma(\cdot)$ is the gamma function. Thus:

$$\eta = -\frac{\alpha}{m}\left\|\tilde{\nabla} g(X_k)\right\|_2 \int_0^{\pi/2} \cos(\varphi)\sin^{n-2}(\varphi)\,\mathrm{d}\varphi$$
$$= \frac{2\alpha B_n}{m(n-1)}\left\|\tilde{\nabla} g(X_k)\right\|_2.$$

The η function decreases rapidly for large values of n, remaining insensitive to changes in m but is highly influenced by the value of alpha (**Figure 7**).

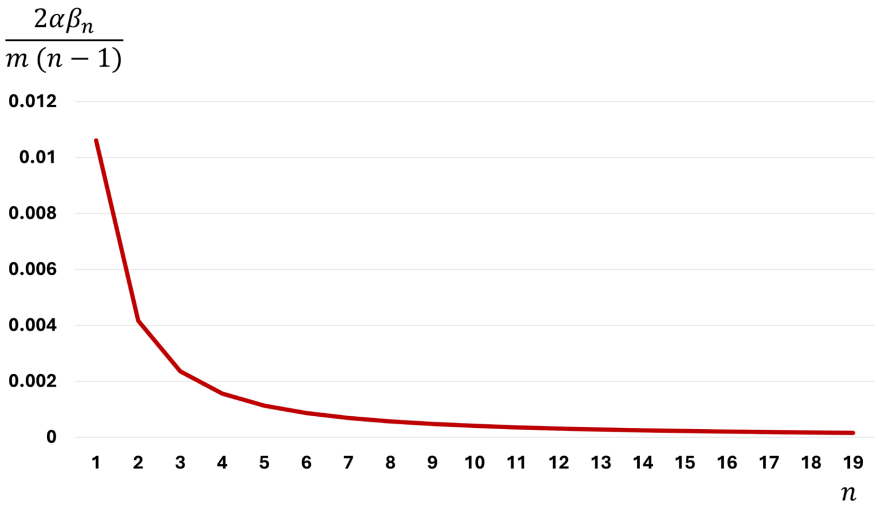

Figure 7. Speed of decline of the efficiency function as a function of n.

Another promising line is to find new ways to construct the subgradients that define the search direction of the heuristic. Here, the technique's competitive advantage lies in the convexity of $g(X)$. Experimenting with various subgradients and step sizes is also an option to improve the method's efficiency.

There are extensive studies on the efficiency of search methods that point to the magnitude of the complexity using this approach [24]; however, this framework constitutes a fascinating alternative for study due to the almost unpredictable nature of the method.

Comparative Aspects of the Alternative Method

Comparing two or more algorithms to evaluate their efficiency is an extensive task

that involves several performance-related criteria. Below is an empirical analysis on the efficiency of the proposal and its comparison with other alternatives, the analysis is based on the suggestions given in [25].

In general, given two algorithms, $AL1$ and $AL2$, $AL1$ is said to be more efficient than $AL2$ if the following relationship holds.

$$\frac{\eta_{AL1}}{\eta_{AL2}} > 1$$

Likewise, the variance of the magnitude Δg_k constitutes another criterion for metric algorithm efficiency.

In the proposal presented in this document, the heuristic requires two initial feasible solutions to trigger the first point of the process using a estimator. The process then progresses by perturbing some of the components of the points already evaluated, retaining the best values of the objective function and eliminating those that do not contribute to minimizing it.

This approach has the advantage of providing great numerical stability, allowing the objective function value to be reduced via feasible solutions at each iteration.

Regarding solution quality, the results suggest a good approximation at the beginning of the search, with the decline slowing as the method approaches the optimal solution. For practical purposes, its implementation is relatively simple because once the first two values of the method are obtained and the first point is reached by approximation, the rest of the process consists of perturbing the components of the last point using a random search process. This is the part of the method that takes the most time because, according to the graph, after four points on the sphere, the efficiency decreases significantly.

Table 3 shows the results obtained when comparing our heuristic (which we present as *OH*) with *LINGO*, *AMPL*, and *GAMS*. The following comparative models (including Model 1, described above) were used to perform these runs [26] [27]. The presented mathematical models were coded in the LINGO optimization software (which uses B&B as the default solver) and run on an Apple computer with an M2-pro chip, 16 GHb of memory, and macOS Sequoia.

- **Model 2:** A model for locating warehouses in a logistics system with fixed costs, 3 warehouses, 3 consumption centers with the following data:

1) Capacity: 300, 525, 325.

2) Demand: 100, 200, 125, 225.

3) Variable costs:

10	5	12	3
4	9	15	6
15	8	6	11

4) Fixed costs: 125,000, 185,000, 100,000.

- **Model 3:** A multimodal transport model with 3 origins, 4 destinations and 2 modes of transport.

Table 3. Comparisons of the exact method versus the heuristic.

Model	Running using *LINGO*					Running using *OH*				
	NV	*NC*	*RI*	*OG*	*CPU*	*NV*	*NV*	*RI*	*OG*	*CPU*
1	30	26	63	−1816	0.09	30	26	122	−1816	0.45
2	24	37	28	289,100	0.05	24	37	75	289,000	0.42
3	288	55	9	3425	0.05	24	37	38	3400	0.40
4	3	3	20	0.9808	0.34	3	3	32	0.9800	1.03
5	18	13	242	324,760	0.07	18	13	325	32450	1.27
6	65	72	147	5732	1.03	65	72	290	5730	1.63
Model	Running using *AMPL*					Running using *GAMS*				
	NV	*NC*	*RI*	*OG*	*CPU*	*NV*	*NV*	*RI*	*OG*	*CPU*
1	30	26	80	−1816	0.11	30	26	80	−1816	0.24
2	24	37	32	289,100	0.09	24	37	33	289,000	0.14
3	288	55	14	3425	0.06	288	55	15	3400	0.11
4	3	3	24	0.9808	0.38	3	3	24	0.9800	0.39
5	18	13	248	324,760	1.01	18	13	250	324,760	1.00
6	65	72	152	5732	1.07	65	72	152	5732	1.08

Where *NV* denotes the number of variables involved, *NC* is the number of model constraints, *RI* represents the number of iterations required for convergence, *OG* is the global optimal value, *CPU* is the time in seconds required by the computer.

1) Capacity: 200, 150, 300.

2) Demand: 100, 200, 125, 225.

3) Variable costs:

10.8	5.4	12.14	3.4
4.6	9.8	15.12	6.5
15.17	8.9	6.9	11.8

- **Model 4:** A reliability nonlinear model

$$\text{Maximize } R=\left(1-\left(1-0.65\right)^{4d+1}\right)\left(1-\left(1-0.55\right)^{3+d_2}\right)\left(1-\left(1-0.70\right)^{4+d_3}\right)$$

Subject to:

$$16d_1+12d_2+13d_3\le 75,\ 2\le d_1\le 3,\ 2\le d_2\le 2,\ 2\le d_3\le 4$$

$$d_1,d_2,d_3\in\mathcal{Z}^+$$

- **Model 5:** A production planning model with multiple processes and multiple products [28].

$$\text{Minimize}\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{m_j}\left(C_{ijt}^{p}P_{ijt}+C_{it}^{I}I_{it}\right)$$

Subject to

$$\sum_{i=1}^{N}\sum_{j=1}^{m_j}a_{ijk}P_{ijt}\le A_{kt},\ t=1,2,\cdots,T;\ k=1,2,\cdots,K$$

$$I_{it} = I_{it-1} + \sum_{j=1}^{m_j} P_{ijt} - D_{it},\ t = 1,2,\cdots,T;\ i = 1,2,\cdots,N$$

$$P_{ijt} I_{it} \in \mathcal{Z}^{+},\ t = 1,2,\cdots,T;\ i = 1,2,\cdots,N;\ j = 1,2,\cdots,m_j$$

With the following instance,

$$\begin{aligned}\text{Minimize } g(X) = {} & 5I_{11} + 6I_{12} + 6I_{21} + 7I_{22} + 7I_{23} + 72P_{111} + 80P_{121} + 85P_{211} \\ & + 90P_{221} + 74P_{112} + 78P_{122} + 88P_{212} + 95P_{222} + 75P_{113} \\ & + 78P_{123} + 4P_{213} + 92P_{223}\end{aligned}$$

Subject to:

$$\begin{aligned}& 5p_{111} + 4p_{121} + 8p_{211} + 6p_{221} \le 8600 \\ & 10p_{111} + 8p_{121} + 12p_{211} + 9p_{221} \le 17000 \\ & 5p_{112} + 4p_{122} + 8p_{212} + 6p_{222} \le 8500 \\ & 10p_{112} + 8p_{122} + 12p_{212} + 9p_{222} \le 16600 \\ & 5p_{113} + 4p_{123} + 8p_{213} + 6p_{223} \le 8800 \\ & 10_{p113} + 8p_{123} + 12p_{213} + 9p_{223} \le 18200 \\ & I_{11} = 100 + P_{111} + P_{121} - 1000 \\ & I_{12} = I_{11} + p_{112} + p_{122} - 1050 \\ & I_{13} = I_{12} + P_{113} + P_{123} - 1100 \\ & I_{21} = 50 + P_{211} + P_{221} - 500 \\ & I_{22} = I_{21} + P_{212} + P_{222} - 600 \\ & I_{23} = I_{22} + P_{213} + P_{223} - 550\end{aligned}$$

$$P_{ijt} I_{it} \in \mathcal{Z}^{+}$$

- **Model 6:** A vehicle routing model visiting eight cities with a load capacity of 18 tons. The following distance and demand matrix is as follows.

1) Demand: 0, 6, 3, 7, 7, 18, 4, 5,

2) Dist. matrix:

0	996	2162	1067	499	2054	2134	2050
0	0	1167	1019	596	1059	1227	1055
0	1167	0	1747	1723	214	168	250
0	1019	1747	0	710	1538	1904	1528
0	596	1723	710	0	1589	1827	1579
0	1059	214	1538	1589	0	371	36
0	1227	168	1904	1827	371	0	407
0	1055	250	1528	1579	36	407	0

Statistical Analysis

In order to develop comparative statistics of the execution times of the proposed models, the results obtained by applying three non-parametric tests designed specifically for such purposes are shown below [29].

- **Friedman Test**: As a first test, the non-parametric Friedman test is used to compare related groups since the data do not meet the assumptions of normality.

The Friedman test is used to demonstrate that there are significant differences in the sample data in Table 3. The null hypothesis indicates that all algorithms behave similarly. Table 4 shows the ranges obtained in relation to the execution time variable.

Table 4. Range analysis for the Friedman statistic.

LINGO	*OH*	*AMPL*	*GAMS*
0.09 (1)	0.45 (4)	0.11 (2)	0.24 (3)
0.05 (1)	0.42 (4)	0.09 (2)	0.14 (3)
0.05 (1)	0.40 (4)	0.06 (2)	0.11 (3)
0.34 (1)	1.03 (4)	0.38 (2)	0.39 (3)
0.07 (1)	1.27 (4)	1.01 (3)	1 (2)
0.03 (1)	1.63 (4)	1.07 (2)	1.08 (3)

Then, for $n=6$ and $k=3$, we have that the Friedman statistic is equal to

$$F_F = \frac{12}{nk(k+1)}\left[\sum_j R_j^2 - 3n(k+1)\right] = 166.26 \tag{28}$$

Thus, for a critical value of 0.10, $F_F = 4.60$ and the null hypothesis is rejected and there are large significant differences between the running times of the algorithms.

- **Multiple Sign Test**: This test uses LINGO as a study control method. This test is an extension of the traditional method. Its objective is to determine the direction of the differences (signs) rather than their magnitude. The results are shown in Table 5.

Table 5. Comparative results for the sign test.

LINGO	OH	AMPL	GAMS
0.09	+	+	+
0.05	+	+	+
0.05	+	+	+
0.34	+	+	+
0.07	+	+	+
0.03	+	+	+
Number of minus signs	6	6	6
Number of plus signs	0	0	0
r_j	0	0	0

Where the respective medians of each method are given by: $Med_{Lingo} = 0.08$, $Med_{OH} = 0.74$, $Med_{AMPL} = 0.245$ and $Med_{GAMS} = 0.31$. Thus, for an $\alpha = 0.05$, the hypothesis $H_0 : M_{Lingo} \leq M_j$ is accepted since the number of plus signs is less

than the critical value $\mathcal{R}=6$, with $\rho=24$ and $k-1=3$.

- **Kruskal-Wallis Test**: The Kruskal-Wallis test is a non-parametric test that will be used to compare means and medians of independent groups, considering the time taken by each algorithm to solve the proposed instances as a variable of interest. This test uses the following statistic to maintain the null hypothesis of equality between the average times for all the algorithms tested (Table 3). Table 6 shows the summary of calculations for this test.

$$K_W=\left(\frac{\rho-1}{\rho}\right)\sum_{i=1}^{k}\frac{\rho_i\left[\bar{R}_i-\mathbf{E}[R]\right]^2}{\sigma^2} \tag{29}$$

where $\bar{R}_L=6.4166$, $\bar{R}_A=10$, $\bar{R}_O=16.666$, $\bar{R}_G=11.2500$, $E[R]=\frac{\rho+1}{2}=12.5$ and $\sigma_R^2=\frac{\rho^2-1}{12}=52$. Thus, for $\rho=\sum_{i=1}^{24}\rho_i=24$, we get that $K_W=8.8692$. Thus, for $\chi^2_{0.05,3}=7.8147$ the null hypothesis is rejected.

Table 6. Ranges associated with the times used by the algorithms.

Alg.	*L*	*L*	*A*	*L*	*L*	*A*	*G*	*A*	*G*	*G*	*L*	*A*
Time	0.05	0.05	0.06	0.07	0.09	0.09	0.11	0.11	0.14	0.24	0.34	0.38
R	1.5	1.5	3	4	5.5	5.5	6.5	6.5	7	8	9	10
Alg.	*G*	*O*	*O*	*O*	*G*	*A*	*L*	*O*	*A*	*G*	*O*	*O*
Time	0.39	0.40	0.42	0.45	1	1.01	1.03	1.03	1.07	1.08	1.27	1.63
R	11	12	13	14	15	16	17	18	19	20	21	22

From the above results, it can be inferred that, although the proposed heuristic appears promising, its computational efficiency is currently low compared to the most common software programs on the market. Therefore, research into how to improve the perturbation direction of the vector X_k would have a strong impact on the speed of convergence. Finally, converting the heuristic into a metaheuristic would be the ideal alternative for implementation in engineering models that frequently involve millions of integer variables to address real-world problems.

Like any heuristic, our proposal almost always provides convergent solutions; however, in comparison, it is not very competitive with the branch-and-bound method for cases with few variables. However, as mentioned at the beginning of this presentation, this proposal is an alternative approach to traditional techniques, such as cutting planes. As part of future research, it is recommended to evaluate larger instances and refine the perturbation method to improve search efficiency.

5. Conclusions and Suggestions

This article presents an alternative heuristic to traditional scientific methods for optimizing an integer linear programming model.

The method consists of proposing two initial feasible points and, from there, us-

ing a quasi-gradient search method to determine the direction of movement and regulate the step size to remain within the set of integer values.

The novelty of this proposal lies in the fact that the method begins its search within the feasible region and remains there, selecting movement directions by means of slight perturbations to the components of the points that are not feasible, always searching for a downward direction.

The results suggest the viability of the method for use in large-scale models since, despite the enormous computational effort required to locate new points in the sequence, the search for a downward direction becomes rapid at the beginning of the process, but slows down as the method approaches the optimal solution or when the support point in iteration k is on the boundary of the set $\mathcal{D}$.

The results shown for locating points on a hypersphere suggest that a maximum of four perturbed components maintains a good efficiency for the method. Although this result is not conclusive, the alternative is to suggest other types of surfaces for searching for alternate candidates in the descent sequence.

The experience gained in this research suggests expanding research into construction methods, search step sizes, and the selection of the best candidate components to be perturbed during the process. Future avenues in this regard remain open to provide more efficient methods based on the aforementioned conditions.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Ermolieva, T., Ermoliev, Y., Obersteiner, M. and Rovenskaya, E. (2021) Chapter 4 Two-Stage Nonsmooth Stochastic Optimization and Iterative Stochastic Quasigradient Procedure for Robust Estimation, Machine Learning and Decision Making. In: Roberts, F.S. and Sheremet, I.A., Eds., Resilience in the Digital Age, Springer, 45-74. https://doi.org/10.1007/978-3-030-70370-7_4

[2] Pérez Lechuga, G. (2018) Optimal Logistics Strategy to Distribute Medicines in Clinics and Hospitals. *Journal of Mathematics in Industry*, **8**, Article No. 2. https://doi.org/10.1186/s13362-018-0044-5

[3] Pérez-Lechuga, G., Aguilar-Velázquez, S.L., Cisneros-López, M.A. and Martínez, F.V. (2019) A Model for the Location and Scheduling of the Operation of Second-Generation Ethanol Biorefineries. *Journal of Mathematics in Industry*, **9**, Article No. 3. https://doi.org/10.1186/s13362-019-0060-0

[4] Pérez-Lechuga, G., Venegas-Martínez, F. and Martínez-Sánchez, J.F. (2021) Mathematical Modeling of Manufacturing Lines with Distribution by Process: A Markov Chain Approach. *Mathematics*, **9**, Article 3269. https://doi.org/10.3390/math9243269

[5] Pérez-Lechuga, G., Venegas-Martínez, F., Montufar-Benítez, M.A. and Mora-Vargas, J. (2022) On the Dynamics in Decoupling Buffers in Mass Manufacturing Lines: A Stochastic Approach. *Mathematics*, **10**, Article 1686. https://doi.org/10.3390/math10101686

[6] Pérez-Lechuga, G., Martínez-Sánchez, J.F., Venegas-Martínez, F. and Madrid-Fernández, K.N. (2024) A Routing Model for the Distribution of Perishable Food in a Green Cold Chain. *Mathematics*, **12**, Article 332. https://doi.org/10.3390/math12020332

[7] Papadimitriou, C.H. (1981) On the Complexity of Integer Programming. *Journal of the ACM*, **28**, 765-768. https://doi.org/10.1145/322276.322287

[8] Rothberg, E. (2007) An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions. *INFORMS Journal on Computing*, **19**, 534-541. https://doi.org/10.1287/ijoc.1060.0189

[9] Fischetti, M. and Lodi, A. (2010) Heuristics in Mixed Integer Programming. In: James, J., Ed., *Wiley Encyclopedia of Operations Research and Management Science*, John Wiley & Sons, Inc, 1-6. https://homepages.cwi.nl/~dadush/workshop/discrepancy-ip/papers/heuristics-survey-fischetti-lodi-11.pdf

[10] Kleinert, T., Labbé, M., Ljubić, I. and Schmidt, M. (2021) A Survey on Mixed-Integer Programming Techniques in Bilevel Optimization. *EURO Journal on Computational Optimization*, **9**, Article ID: 100007. https://doi.org/10.1016/j.ejco.2021.100007

[11] Huang, L.Y., Chen, X.M., Huo, W., Wang, J.Z., Zhang, F., Bai, B. and Shi, L. (2021) Branch and Bound in Mixed Integer Linear Programming Problems: A Survey of Techniques and Trends. arXiv: 2111.06257. https://doi.org/10.48550/arXiv.2111.06257

[12] Ermoliev, Y.M. and Gaivoronski, A.A. (1992) Stochastic Quasigradient Methods for Optimization of Discrete Event Systems. *Annals of Operations Research*, **39**, 1-39. https://doi.org/10.1007/bf02060934

[13] Pérez-Lechuga, G. (1993) bibinfotitleUn algoritmo para la optimización estocástica de algunos modelos dinámicos. Ph.D. Thesis, Universidad Nacional Autónoma de México.

[14] Ermol'ev, Y.M. (1972) On the Method of Generalized Stochastic Gradients and Quasi-Féjer Sequences. *Cybernetics*, **5**, 208-220. https://doi.org/10.1007/bf01071091

[15] Ball, M.O. (2011) Heuristics Based on Mathematical Programming. *Surveys in Operations Research and Management Science*, **16,** 21-38. https://www.researchgate.net/publication/229415600

[16] Borne, P., Popescu, D., Filip, F.G. and Stefanoiu, D. (2014) Optimization in Engineering Sciences. John Wiley and Sons, 1-30.

[17] Combettes, P.L. (2001) Quasi-Fejérian Analysis of Some Optimization Algorithms. *Studies in Computational Mathematics*, **8**, 115-152. https://doi.org/10.1016/s1570-579x(01)80010-0

[18] Boyd, S., Duchi, J., Pilanci, M. and Vandenberghe, L. (2022) Subgradients. https://web.stanford.edu/class/ee364b/lectures/subgradients_notes.pdf

[19] Ermoliev, Y.M. (2025) Stochastic Quasigradient Methods and their Application in Systems Optimization. https://pure.iiasa.ac.at/id/eprint/1759/7/WP-81-002.pdf

[20] Rubinstein, Y. and Reuben, L. (1981) Simulation and the Monte Carlo Method. John Wiley & Sons, Inc.

[21] Svaiter, B.F. (2025) Fejer-Convergent Algorithms Which Accept Summable Errors, Approximated Resolvents and the Hybrid Proximal-Extragradient Method. https://webdoc.sub.gwdg.de/ebook/serien/e/IMPA_A/715.pdf

[22] LINGO (2025) Software for Mathematical Optimization. Integer Programming, Linear Programming, Nonlinear Programming, Stochastic Programming, Global Optmization. https://www.lindo.com/

[23] Rubinstein, R.Y. (1982) Generating Random Vectors Uniformly Distributed inside and on the Surface of Different Regions. *European Journal of Operational Research*, **10**, 205-209. https://doi.org/10.1016/0377-2217(82)90161-8

[24] Pérez-Lechuga, G., Tuoh-Mora, J., Morales-Sánchez, E. and Suárez-Álvarez, M. (2005) On the Efficiency of a Random Search Method. https://www.researchgate.net/publication/241769436_ON_THE_EFFICIENCY_OF_A_RANDOM_SEARCH_METHOD

[25] (2025) How to Compare Two Algorithms Empirically? https://www.baeldung.com/cs/compare-algorithms-performance#::text=Choosing

[26] Liberatore, M. and Nydick, R. (2003) Decision Technology: Modeling, Soft-Ware, and Applications. John Wiley & Sons, Inc.

[27] Gupta, N. and Ali, I. (2021). Optimization with LINGO-18 Problems and Applications. CRC Press. https://doi.org/10.1201/9781003048893

[28] Sipper, D. and Bulfin, R. (1997) Production: Planning, Control, and Integration. McGraw-Hill College.

[29] García, S., Fernández, A., Luengo, J. and Herrera, F. (2010) Advanced Nonparametric Tests for Multiple Comparisons in the Design of Experiments in Computational Intelligence and Data Mining: Experimental Analysis of Power. *Information Sciences*, **180**, 2044-2064. https://doi.org/10.1016/j.ins.2009.12.010

Appendix: Convergence of Solutions of the Proposed Method

Table A1 and Table A2 show the initial and final results of the process. Convergence is achieved after 121 iterations.

Table A1. Initial solution and first five iterations of the process.

$g(X_0)=2069$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$g(X_0)=2022$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	2	3	4	5	6	6	7	8	10	11	11	13	14	15
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
15	17	17	18	19	21	22	22	24	24	25	26	27	29	29
$g(X_0)=1936$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
$k=4,\ g(\hat{X}_k)=1932$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	2	3	3	4	5	6	6	8	10	11	10	12	14	15
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
15	16	16	18	19	20	22	21	24	24	24	25	26	28	29
$k=5,\ g(\hat{X}_k)=1927$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	2	3	3	4	5	6	6	8	10	11	10	12	14	15
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
15	16	16	18	19	20	22	21	24	24	24	25	26	28	29
$k=5,\ g(\hat{X}_k)=1926$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
0	7	8	7	4	7	6	6	7	8	14	10	14	12	16
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
15	16	14	18	19	21	22	21	27	23	24	32	26	28	32
$k=5,\ g(\hat{X}_k)=1905$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
0	2	2	2	4	4	6	6	9	10	11	9	11	13	17
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
14	16	16	14	19	19	21	20	23	23	24	22	25	28	29

Table A2. Last seven iterations of the process.

$g(X_0)=-1782$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	6	8	4	0	0	22	0	0	0	16	0	0	0	0
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
20	22	26	24	0	0	0	0	136	0	0	0	0	45	75
$g(X_0)=-1790$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	6	8	3	0	0	22	0	0	0	16	0	0	0	0
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
20	22	26	24	0	0	0	0	136	0	0	0	0	45	75
$g(X_0)=-1798$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	6	8	2	0	0	22	0	0	0	16	0	0	0	0
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
20	22	26	24	0	0	0	0	136	0	0	0	0	45	75
$g(X_0)=-1806$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	6	8	1	0	0	22	0	0	0	16	0	0	0	0
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
20	22	26	24	0	0	0	0	136	0	0	0	0	45	75
$g(X_0)=-1808$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	6	7	0	0	0	22	0	0	0	16	0	0	0	0
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
20	22	26	24	0	0	0	0	136	0	0	0	0	45	75
$g(X_0)=-1814$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	6	8	0	0	0	22	0	0	0	16	0	0	0	0
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
20	22	26	24	0	0	0	0	136	0	0	0	0	45	75
$g(X_0)=-1816$														
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
0	6	8	0	0	0	22	0	0	0	16	0	0	0	0
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}
20	22	26	24	0	0	0	0	136	0	0	0	0	45	75