

Article

Generative Heuristics for Hard- and Soft-Criteria Optimization: Integrating LLMs Within Metaheuristics

Miguel Saiz ¹, Angel A. Juan ^{2,3,*} and Javier Panadero ⁴

¹ Operations, Innovation and Technology Management Department, ESADE Business School, 08034 Barcelona, Spain; miguel.saiz@esade.edu

² CIGIP—ValgrAI, Universitat Politècnica de València, 03801 Alcoy, Spain

³ Data Analytics Department, UNIE Universidad, Av. Monforte de Lemos 28, 28029 Madrid, Spain

⁴ Computer Architecture & OS Department, Universitat Autònoma de Barcelona, 08193 Barcelona, Spain; javier.panadero@uab.cat

* Correspondence: ajuanp@upv.es

Abstract

In many real-world domains, optimization problems involve both quantitative objectives and qualitative criteria that cannot be directly formulated as mathematical functions. While recent research has integrated large language models (LLMs) into optimization algorithms, these approaches primarily use LLMs to guide the search process, generate heuristics, or tune optimization strategies, rather than to evaluate semantic objectives. This paper introduces generative heuristics, a methodology that combines traditional metaheuristics with LLM-based semantic evaluation to address ‘soft optimization’ problems containing both hard quantitative constraints and soft qualitative objectives. The proposed methodology first applies a metaheuristic algorithm to generate a shortlist of high-quality candidate solutions satisfying the quantitative objectives. These candidates are then evaluated by an LLM according to a user-defined qualitative rubric, and the resulting semantic scores are incorporated into the optimization process through a weighted soft objective. The methodology is illustrated through a constrained portfolio optimization problem in which financial risk is minimized while simultaneously promoting strategic objectives related to sector diversification, geographic exposure, environmental-social-governance quality, and overall portfolio coherence. Experimental results show that the proposed methodology consistently improves strategic alignment while producing solutions that remain close to those obtained from the hard optimization formulation.

Keywords: soft optimization; metaheuristics; large language models; decision support systems; generative heuristics



MSC: 68T20; 90-08; 90-10; 90B50

Academic Editors: Hung-Lung Lin and Yu-Yu Ma

Received: 14 June 2026

Revised: 9 July 2026

Accepted: 13 July 2026

Published: 15 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Soft optimization refers to a class of decision problems in multi-criteria optimization where objectives and constraints include both hard (quantitative) and soft (qualitative) components. In traditional multi-criteria optimization, these are often handled via weighted sum methods or Pareto-frontier analysis. However, hard elements, such as cost minimization, profit maximization, or feasibility constraints, are modeled using mathematical programming techniques based on well-defined numerical functions. In contrast, soft components, including strategic alignment, stakeholder preferences, ethical considerations,

or policy compliance, are difficult to represent in standard multi-criteria models because they depend on contextual interpretation and semantic understanding, which are not easily captured in conventional mathematical formulations. This limitation is evident in many real-world applications where decisions must balance measurable performance with intangible values. In project portfolio management, for example, selecting projects that maximize expected value while aligning with organizational strategy requires evaluating qualitative criteria such as innovation potential or sustainability impact [1,2]. Similarly, in supply chain design, supplier selection involves not only cost and lead time but also environmental and social considerations, which are often expressed in textual or policy-based form [3]. Existing optimization approaches struggle to integrate these dimensions within a unified and operational manner.

In effect, traditional methods remain largely focused on quantitative metrics and do not incorporate semantic reasoning into the decision loop. Studies in machine learning, and in particular natural language processing, have improved the ability to extract structured signals from unstructured text [4,5]. However, most approaches treat these signals as inputs to predictive models rather than embedding language understanding directly into the optimization process. Some studies on LLMs have shown their capacity to perform complex reasoning and contextual evaluation tasks [6], yet their integration into formal optimization pipelines remains limited. This gap motivates the development of hybrid methods that combine search-based optimization with semantic reasoning. This paper introduces a hybrid methodology that integrates metaheuristic search with LLM-based semantic evaluation. In this framework, the metaheuristic component explores the feasible solution space under hard constraints, such as budget or resource limits, using established techniques [7,8]. In parallel, the LLM processes qualitative inputs, such as strategic documents or project descriptions, to assess semantic alignment with soft objectives and to generate new candidate solutions that reflect contextual goals. A key feature of the proposed approach is the feedback loop between these components. Candidate solutions generated or modified by the metaheuristic are evaluated not only on quantitative criteria but also on semantic relevance. Conversely, the LLM proposes new configurations that are injected into the search process, guiding exploration toward regions of the solution space that better satisfy qualitative objectives. This interaction enables the system to iteratively converge toward solutions that balance financial performance with alignment to abstract goals such as sustainability, equity, or organizational vision (Figure 1).

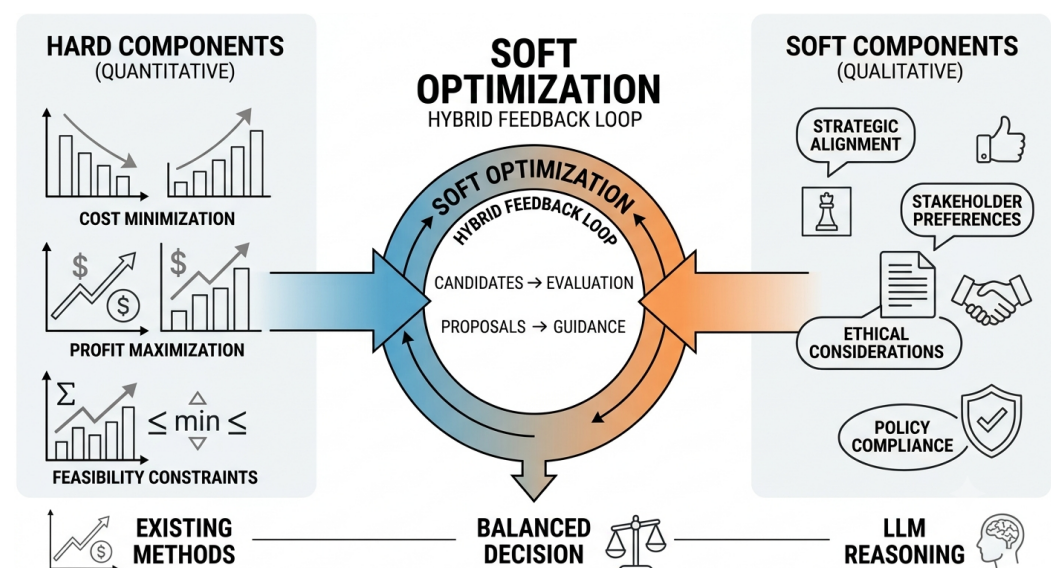


Figure 1. LLM-support for multi-criteria soft optimization problems.

To formalize this interaction, let $\mathcal{X}$ denote the feasible solution space and $x \in \mathcal{X}$ represent a candidate solution. The optimization problem is defined by a composite objective function $F(x) = f_{\text{hard}}(x) + \lambda f_{\text{soft}}(x)$, where $f_{\text{hard}}(x)$ quantifies measurable performance metrics and $f_{\text{soft}}(x)$ captures qualitative alignment. The soft component is derived via a semantic mapping process: a candidate x is first transformed into a structured textual representation $\phi(x)$ using context c . An LLM then evaluates this representation to produce a semantic assessment $s = \text{LLM}(\phi(x), c)$, which is subsequently mapped to a scalar score via a function $\phi(\cdot)$, yielding $f_{\text{soft}}(x) = \phi(s)$. In the proposed generative heuristic framework, the search process $\mathcal{S}$ iteratively generates a candidate pool $\mathcal{C}_t$ through two parallel mechanisms: stochastic exploration using heuristic operators, denoted as $\mathcal{C}_t^{\text{meta}} = \mathcal{S}_{\text{meta}}(\mathcal{P}_t)$, and semantic generation conditioned on the current best solutions $\mathcal{P}_t^*$, denoted as $\mathcal{C}_t^{\text{LLM}} = \text{LLM}_{\text{gen}}(c, \mathcal{P}_t^*)$. The unified pool $\mathcal{C}_t = \mathcal{C}_t^{\text{meta}} \cup \mathcal{C}_t^{\text{LLM}}$ is then evaluated using $F(x)$, and the population $\mathcal{P}_{t+1}$ is updated to guide the search toward regions that satisfy both hard constraints and soft strategic objectives. In order to illustrate the proposed approach, a case study is employed to show that the integration of metaheuristics and LLMs can improve solution quality in optimization problems characterized by both hard and soft criteria. Specifically, the classical mean-variance portfolio selection problem introduced by Markowitz [9] is considered. This model defines the efficient frontier. This paper extends the original model within a soft optimization context by allowing qualitative preferences to complement the traditional risk–return trade-off. The results indicate that the proposed approach can improve performance in decision settings where purely quantitative methods may be insufficient.

The main contributions of this paper are threefold: (i) it introduces ‘soft optimization’ as a distinct class of problems where qualitative constraints require semantic reasoning; (ii) it proposes a ‘generative heuristic’ framework that unifies heuristic search with LLM-based semantic evaluation and generation; and (iii) it illustrates the efficacy of the proposed methodology through a portfolio optimization case study, showing improved alignment with strategic objectives without significant loss in financial performance. The rest of this paper is structured as follows: Section 2 reviews related work on hybrid optimization methods, the integration of LLMs in decision support systems, and existing models for soft objectives; Section 3 presents the proposed framework, detailing how metaheuristics and LLMs are integrated within a unified search-semantic loop; Section 4 discusses key computational considerations including algorithmic setup, parameter configurations, and efficiency analysis; Sections 5–7 detail a case study with problem formulations, implementation steps, and results; Section 8 provides verification and statistical analysis; Section 9 reviews scalability and cost aspects of the soft-optimization layer; finally, Section 10 provides managerial insights into the practical application of this framework in real-world settings, followed by concluding remarks in Section 11.

2. Related Work

Metaheuristic algorithms have been widely used to address complex optimization problems where exact methods are not tractable [10]. These approaches efficiently explore large and non-convex search spaces and have been successfully applied to engineering design, logistics, scheduling, and many other optimization domains [8,11,12]. In particular, metaheuristics have shown strong performance in multi-objective and constrained optimization problems [13]. Despite their success, traditional metaheuristics operate exclusively on quantitative representations of the problem and therefore lack mechanisms to directly incorporate qualitative or semantic information. The integration of qualitative criteria has instead been studied within multi-criteria decision-making (MCDM) frameworks, which combine quantitative and qualitative factors through scoring, weighting, or pairwise

comparison methods [14,15]. Applications include project portfolio selection and sustainable supply chain management, where non-financial criteria are important [16]. However, these approaches generally require qualitative judgments to be manually translated into numerical scores and are typically applied after the optimization process rather than being embedded within it.

Some studies on LLMs have shown strong capabilities in natural language understanding and semantic reasoning [4,5]. These capabilities have motivated growing interest in combining LLMs with optimization algorithms. Existing work can be broadly grouped into three directions. The first uses LLMs to guide optimization by generating search strategies, adapting operators, or assisting the optimization process [17,18]. The second employs LLMs as hyper-heuristics, where language models generate, select, or adapt heuristics for solving optimization problems [19,20]. The third investigates the interaction between LLMs and evolutionary computation, including LLM-assisted evolutionary algorithms, multi-agent optimization frameworks, and the broader integration of foundation models with evolutionary computation [21–25]. Although these studies illustrate the potential of combining LLMs with optimization, their primary objective is to improve the optimization process itself by designing better search operators, generating heuristics, tuning algorithm parameters, or accelerating convergence. In these approaches, the LLM mainly supports how the search is conducted rather than what is being optimized. Other studies have further expanded the integration of LLMs into optimization workflows. OptiMUS [26] shown that LLMs can automatically translate optimization problems expressed in natural language into mathematical programming models solvable by mixed-integer optimization solvers, while OptiMUS-0.3 [27] extended this approach to improve scalability and robustness for larger optimization instances. Another proposed approach is ORLM [28], a customizable framework for training LLMs specifically for optimization modeling. It shows that domain-specific fine-tuning substantially improves the quality of automatically generated optimization formulations compared with prompt engineering alone. Beyond optimization modeling, recent work has explored the automatic design of optimization heuristics using LLMs [29], benchmarked their reasoning capabilities on constrained combinatorial optimization problems expressed in natural language, and introduced infeasibility-aware frameworks that enable LLMs to identify infeasible solutions while generating high-quality initial candidates for downstream optimization. These innovations illustrate the rapid evolution of LLM-assisted optimization, spanning optimization modeling, heuristic generation, constrained reasoning, and hybrid optimization strategies.

The methodology proposed in this paper assigns the LLM a fundamentally different role. Rather than supporting the search process, the LLM evaluates qualitative objectives that cannot be naturally formulated as explicit mathematical objective functions. The optimization algorithm remains responsible for optimizing the quantitative objectives, while the LLM provides semantic assessments of solution quality according to a user-defined rubric. These assessments are incorporated into the optimization as a soft objective, allowing quantitative and qualitative criteria to be optimized simultaneously. Consequently, the proposed methodology targets a different class of optimization problems in which part of the objective depends on semantic interpretation rather than purely numerical evaluation. To the best of our knowledge, there is still limited work providing a systematic methodology for integrating semantic evaluation directly into the optimization objective, instead of using LLMs only to improve the optimization algorithm itself.

3. Generative Heuristics

The notation used in the general methodological formulation is defined next. Notice that this formulation is intentionally expressed at an abstract level, since the complete portfolio-specific objective function and constraints are introduced later in the case-study section. Let $\mathcal{X}$ denote the solution space containing all candidate solutions, and let $x \in \mathcal{X}$ denote a generic candidate solution. The composite objective function is denoted by $F(x)$, where $f_{\text{hard}}(x)$ represents the quantitative objective component and $f_{\text{soft}}(x)$ represents the soft objective component obtained from the LLM-based evaluation. The parameter λ controls the influence of the soft objective. Hard feasibility constraints are represented by $g_i(x)$, with $i = 1, \dots, m$, where m is the number of such constraints. The context provided to the LLM is denoted by c , and the resulting textual or structured assessment by s . The function $\phi(\cdot)$ maps the LLM response into a numerical score. The set $\mathcal{P}_t^*$ contains the high-quality solutions used to guide LLM-based generation, and x_{new} denotes a new candidate solution generated by the LLM. The following formulation provides the general methodological structure of the proposed soft-optimization framework. It is not intended to replace the problem-specific mathematical model, which is introduced later in the case-study section.

Consider a soft optimization problem defined over a solution space $\mathcal{X}$. Each candidate solution $x \in \mathcal{X}$ is evaluated using both quantitative and qualitative criteria. Quantitative components represent measurable performance indicators, while qualitative components capture preference-based or semantic assessments that require contextual interpretation. A scalarized formulation for a minimization problem can be written as:

$$\min_{x \in \mathcal{X}} F(x) = f_{\text{hard}}(x) + \lambda f_{\text{soft}}(x) \quad (1)$$

subject to hard feasibility constraints:

$$g_i(x) \leq 0, \quad i = 1, 2, \dots, m \quad (2)$$

where λ is a weighting parameter, $f_{\text{hard}}(x)$ denotes the quantitative component of the underlying hard optimization problem, and $f_{\text{soft}}(x)$ denotes a cost derived from qualitative or semantic evaluation. The quantitative component of the underlying hard optimization problem, The hard constraints $g_i(x)$ define feasibility requirements that must be satisfied independently of the soft evaluation. To ensure comparability, both objective components are normalized via min-max scaling to $[0, 1]$ before being combined. If the LLM produces a higher-is-better alignment score, the score is inverted so that higher alignment corresponds to a lower cost, consistent with the minimization objective. In this formulation, the LLM acts as a semantic evaluator that maps a candidate solution into a scalar score reflecting qualitative objectives such as sustainability, equity, or strategic alignment. This score is then integrated into the overall objective function and used within the heuristic search procedure to guide exploration toward solutions that satisfy both quantitative and qualitative criteria.

The methodology proposed in this paper for solving the soft version of the portfolio optimization problem combines a metaheuristic search process with an LLM. First, a genetic algorithm (GA) solves the hard optimization problem (variance minimization) for a grid of target returns. Thus, an initial feasible population is generated and evaluated according to the portfolio variance objective while satisfying the target-return and budget constraints. Each generation applies tournament selection, simulated binary crossover (SBX), and polynomial mutation. Since these operators may generate infeasible portfolios, a repair procedure is applied to restore feasibility by enforcing the minimum-weight, budget, and target-return constraints. The offspring are then re-evaluated and combined

with elitism through a Hall-of-Fame (HoF) mechanism, which preserves the best solutions found during the search. To reduce premature convergence, the algorithm also incorporates adaptive population restarts after prolonged stagnation and warm-start initialization between consecutive target-return levels. The HoF contains several financially high-quality candidate portfolios around each target return, constituting the hard optimization shortlist (HOS). This shortlist is then systematically evaluated using the non-financial criterion in the next stage. The HOS candidates include the chosen asset identifiers, sector, geography, ESG (environmental, social and governance) rating, and the portfolio weight. The LLM prompt is then constructed from three elements: the complete asset metadata, the evaluation rubric, and a batch of HOS candidates. The prompt explicitly instructs the model to ignore financial performance quantities such as return and variance. Thus, the LLM score is used only as a soft strategic signal, while financial feasibility remains defined by the previous hard optimization stage.

To reduce ordering bias, the evaluation stage repeats each prompt batch several times. Odd repeats use the conventional batch order, whereas even repeats reverse both the order of prompt calls and the order of candidates inside each prompt. The stored candidate identifiers remain unchanged, so repeated LLM observations can be aggregated by candidate. The final strategic score used by the parametric lambda study is the mean score across repeated calls, also using the standard deviation as a measure of confidence. Other alternative aggregations (i.e., median) can be considered to minimize effects related with skewness or outliers in the scoring.

Soft components are computed through structured prompts. Given a candidate solution x and a context c , the LLM produces an assessment:

$$s = \text{LLM}(x, c) \quad (3)$$

where s is a textual or structured response. This response is transformed into a numerical score $f_{\text{soft}}(x)$ using a scoring function $\phi(\cdot)$:

$$f_{\text{soft}}(x) = \phi(s) \quad (4)$$

To improve consistency, prompts follow a fixed template and may include evaluation criteria, scoring rubrics, and examples.

In addition to evaluation, the LLM can be used to generate new candidate solutions. Given a context c and a set of high-quality solutions $\mathcal{P}_t^*$, the LLM produces a new solution:

$$x_{\text{new}} = \text{LLM}_{\text{gen}}(c, \mathcal{P}_t^*) \quad (5)$$

Generated solutions are encoded into the same representation used by the metaheuristic. If needed, a repair operator ensures feasibility with respect to hard constraints. The overall procedure is summarized in the block diagram shown in Figure 2, where hard feasibility, qualitative objectives, and soft-constraint preferences are jointly integrated within the search loop.

The interaction between the metaheuristic and the LLM introduces additional design choices. These include the definition of prompt templates, the mapping from textual outputs to numerical scores, and the balance between exploration and exploitation. The weighting parameter λ controls the influence of soft objectives and can be tuned based on the application. The framework is modular. Different metaheuristics or language models can be used without changing the overall structure. This allows the method to adapt to different problem domains and scales. As an example, the proposed methodology is instantiated using a DEAP GA [30], as shown in Algorithm 1. The proposed algorithm

operates as an iterative hybrid loop that alternates between stochastic exploration via genetic operators and semantic exploitation via LLMs. In each generation t , the algorithm first generates a set of candidate solutions $\mathcal{C}_t^{\text{meta}}$ using standard selection, crossover, and mutation operators on the current population $\mathcal{P}_t$. Concurrently, it employs the LLM to generate a set of semantically informed candidates $\mathcal{C}_t^{\text{LLM}}$, conditioned on the problem context c and the current best-known solutions. These two sets are merged into a unified candidate pool $\mathcal{C}_t$. Before evaluation, each candidate in $\mathcal{C}_t$ undergoes a repair phase to ensure feasibility with respect to hard constraints (e.g., budget and weight sum constraints). The core of the framework is the evaluation phase, where each candidate x is assessed using the composite objective function $F(x) = f_{\text{hard}}(x) + \lambda f_{\text{soft}}(x)$. Here, $f_{\text{hard}}(x)$ represents the normalized quantitative cost (e.g., portfolio variance), while $f_{\text{soft}}(x)$ is derived from the LLM's semantic evaluation. To mitigate the stochasticity of LLM outputs, the soft score is computed by aggregating results from multiple prompt evaluations, ensuring a robust estimate of qualitative alignment. Finally, a survivor selection mechanism updates the population $\mathcal{P}_{t+1}$ based on the combined fitness scores, preserving high-performing solutions across both financial and strategic dimensions. The process repeats until the stopping criterion is met, returning the best solution x^* found.

Algorithm 1 Hybrid GA with LLM-based Semantic Evaluation

```

1: Initialize population  $\mathcal{P}_0$  with feasible solutions
2:  $t \leftarrow 0$ 
3: while stopping criterion not met do
4:   // Candidate generation (two branches)
5:   Generate  $\mathcal{C}_t^{\text{meta}}$  from  $\mathcal{P}_t$  using Selection, Crossover, Mutation
6:   Generate  $\mathcal{C}_t^{\text{LLM}}$  using  $\text{LLM}_{\text{gen}}(c, \mathcal{P}_t)$ 
7:   // Merge
8:    $\mathcal{C}_t \leftarrow \mathcal{C}_t^{\text{meta}} \cup \mathcal{C}_t^{\text{LLM}}$ 
9:   // Repair
10:  for each  $x \in \mathcal{C}_t$  do
11:     $x \leftarrow \text{Repair}(x)$ 
12:     $x \leftarrow \text{LocalSearch}(x)$ 
13:  end for
14:  // Evaluation loop (matches diagram)
15:  for each  $x \in \mathcal{C}_t$  do
16:    Compute  $f_{\text{hard}}(x)$ 
17:    Compute  $f_{\text{soft}}(x)$  via LLM
18:    Compute
                                
$$F(x) = f_{\text{hard}}(x) + \lambda f_{\text{soft}}(x)$$

19:  end for
20:  // Population update (GA survival step)
21:   $\mathcal{P}_{t+1} \leftarrow \text{SurvivorSelection}(\mathcal{P}_t, \mathcal{C}_t)$ 
22:   $t \leftarrow t + 1$ 
23: end while
24: return  $x^* \in \mathcal{P}_t$  s.t.  $F(x^*) = \min_{x \in \mathcal{P}_t} F(x)$ 

```

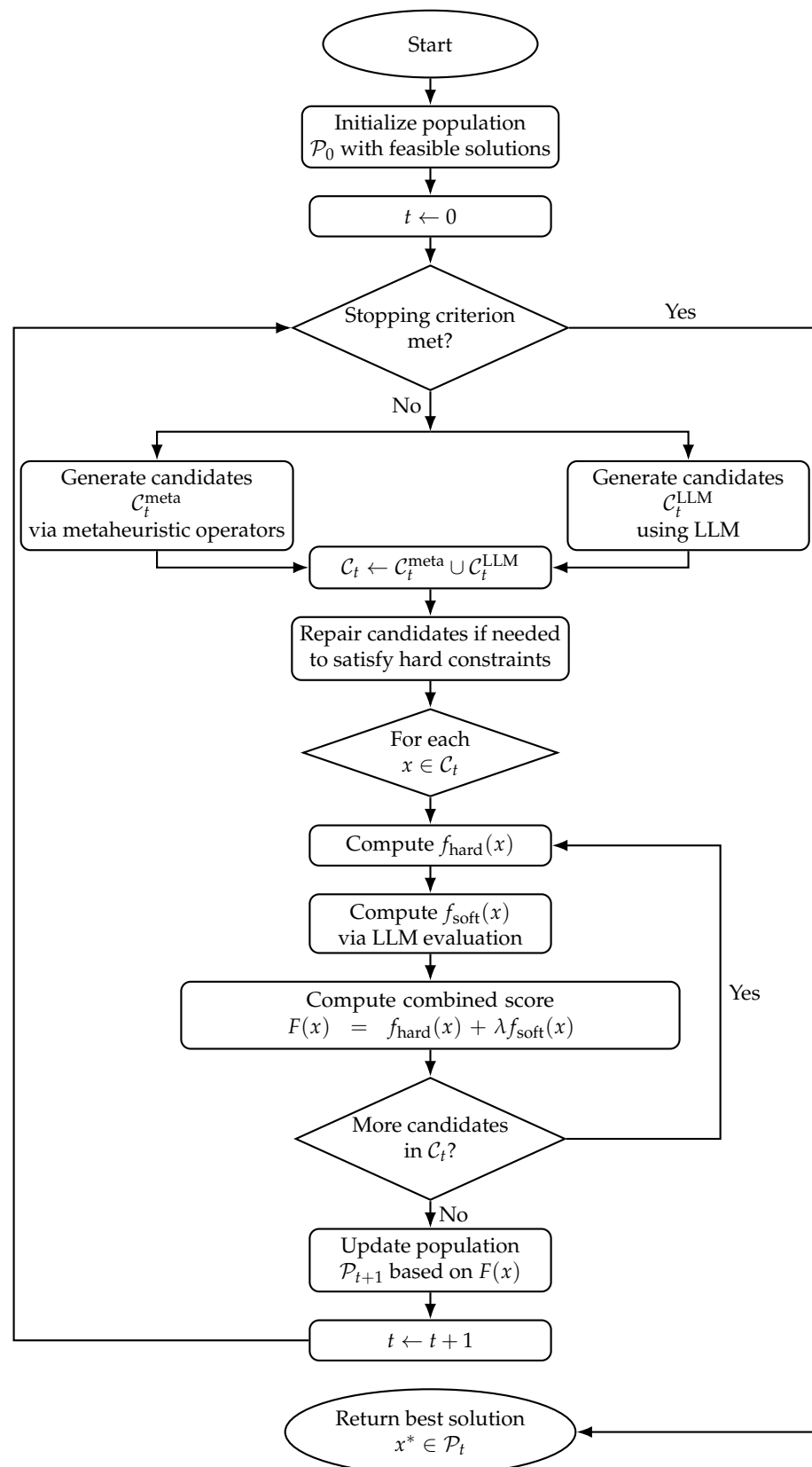


Figure 2. Block diagram for the generative metaheuristic framework.

4. Computational Issues

The proposed methodology introduces additional computational layers compared to standard metaheuristics. In particular, the integration of LLMs raises challenges related to

latency, cost, scalability, and reproducibility. This section discusses key implementation choices and their implications. A first design choice concerns whether the LLM is accessed through an online API or deployed locally. Online LLMs provide access to state-of-the-art models with strong reasoning capabilities and minimal setup effort. However, they introduce non-negligible latency due to network communication and may incur usage costs proportional to the number of queries. In addition, reproducibility can be affected by model updates and limited control over inference settings. Local LLMs, in contrast, offer full control over the model and inference pipeline. They allow for reproducible experiments and eliminate dependency on external services. However, they require significant computational resources, including GPUs or specialized hardware, and may provide lower performance depending on the model size and training quality. In practice, the choice depends on the trade-off between performance, cost, and experimental control.

Since LLM calls are computationally expensive, the number of queries must be carefully managed. A query budget can be enforced per iteration or for the entire optimization process. Several strategies can improve efficiency. First, caching mechanisms can be used to store previous LLM evaluations and avoid redundant queries for identical or similar solutions. Second, approximate evaluation schemes can be introduced, where the LLM is only queried for a subset of promising candidates. Third, batched evaluations can reduce overhead when supported by the deployment environment. Another approach consists of using surrogate models trained to approximate the LLM-based soft score. The LLM is then queried only to update or calibrate the surrogate periodically. This reduces the overall computational burden while preserving alignment with the semantic objective. The quality and consistency of LLM outputs depend on prompt design. Prompts should follow a fixed template and include clear evaluation criteria. Variability in responses can be reduced by controlling decoding parameters such as temperature and by using deterministic settings when available. To further improve stability, multiple LLM evaluations can be aggregated. For instance, the same candidate solution can be evaluated several times and the results averaged. While this increases computational cost, it reduces variance in the LLM-derived soft components and improves robustness of the search process.

The interaction between the metaheuristic and the LLM must be carefully synchronized. Frequent LLM calls may slow down the search, while infrequent calls may limit the influence of semantic guidance. A balance can be achieved by adjusting the frequency of LLM-based generation and evaluation. When soft objectives and soft constraints are assessed separately, additional care is needed to coordinate objective-oriented and constraint-oriented queries. Asynchronous implementations can further improve efficiency. In this setting, LLM queries are executed in parallel with the metaheuristic search. The algorithm continues exploring the solution space while waiting for LLM responses, which are incorporated once available. This reduces idle time and improves resource utilization. Scalability is a critical issue for large problem instances. The cost of evaluating candidates grows with both the population size and the number of LLM queries. To address this, the framework can restrict LLM usage to elite solutions or to selected iterations. Parallelization of the metaheuristic component is also important. Many metaheuristics allow parallel evaluation of candidate solutions, which can be combined with parallel LLM calls when supported. Distributed implementations can further extend the approach to large-scale problems. Reproducibility requires careful control of both the optimization and the LLM components. Random seeds should be fixed for the metaheuristic. For the LLM, prompts, model versions, and inference parameters must be logged. When using online models, it is important to record API versions and timestamps. Storing intermediate solutions, LLM responses, soft-objective scores, and final evaluation scores enables post hoc analysis

and validation. This is particularly relevant when the semantic components depend on stochastic or non-deterministic outputs.

5. Case Study Description

To assess the approach, a portfolio optimization problem over a universe of financial assets is introduced as a case study. In this section, the generic candidate solution x introduced in Section 3 is instantiated as a portfolio. The portfolio-specific notation used in the case study is introduced before presenting the mathematical formulation. Let N denote the set of available assets, with $n = |N|$ representing the number of assets, and let $i \in N$ denote the asset index. The expected return of asset i is denoted by μ_i , and the covariance matrix of asset returns by Σ . The descriptive information associated with asset i , such as its sector, geographical exposure, or ESG profile, is denoted by d_i . The portfolio is represented by the weight vector $w \in \mathbb{R}^n$, where w_i denotes the proportion of capital invested in asset i . The target portfolio return is denoted by ρ , and the LLM strategic alignment score by S . Each asset $i \in \mathcal{N}$ is characterized by an expected return μ_i , a covariance matrix Σ , and descriptive information d_i (e.g., sector, ESG profile, or business activity). The decision variable is a vector of portfolio weights $w \in \mathbb{R}^n$, where w_i represents the proportion of capital invested in asset i . The feasible set satisfies:

$$\sum_{i=1}^n w_i = 1 \quad w_i \geq 0 \quad (6)$$

where the hard objective follows the classical mean-variance framework:

$$\begin{aligned} \min_w \quad & w^\top \Sigma w \\ \text{s.t.} \quad & \mu^\top w \geq \rho \\ & \mathbf{1}^\top w = 1 \\ & w \geq 0 \end{aligned} \quad (7)$$

In addition, the investor provides strategic guidelines expressed in natural language. These include preferences such as sustainability, regional exposure, or diversification. These preferences define a soft objective that measures how well a portfolio aligns with the desired strategy. Given a portfolio x , a textual representation is constructed by aggregating the descriptions of the selected assets (the ones with positive portfolio weights):

$$d(w) = \bigoplus_{i \in N: w_i > 0} d_i \quad (8)$$

The alignment score is computed using an LLM:

$$f_{\text{soft}}(w) = \phi(\text{LLM}(d(w), c)) \quad (9)$$

where c represents the strategic context and $\phi(\cdot)$ maps the LLM response to a numerical score. The prompt includes evaluation criteria such as contribution to sustainability, support for innovation, and consistency with organizational goals. The solution process has two main steps. First, a set of candidate portfolios is generated using a hard optimization method, in this case a GA. The process is repeated for different target configurations, creating a diverse set of feasible and high-quality solutions. A minimum weight constraint is also included to increase diversity. This pool of solutions is referred to as the HOS. Second, the HOS candidates are evaluated using the LLM. This step incorporates soft attributes that are not part of the mathematical model. Each candidate is scored according to a predefined

rubric that reflects the investor's preferences. The LLM also provides structured outputs that explain its assessment.

To improve robustness, the evaluation process is repeated several times with small variations, such as changing the order of inputs. The results are then aggregated statistically. The final fitness of each candidate combines the hard objective and the semantic evaluation:

$$F(x) = f_{\text{hard}}(w) + \lambda f_{\text{soft}}(w) \quad (10)$$

Since the portfolio problem is formulated as a minimization problem, and the LLM returns a strategic alignment score S between 1 and 100, the soft term is defined as:

$$f_{\text{soft}}(w) = 1 - \frac{S}{100} \quad (11)$$

Finally, a parameter sweep over λ is performed. This allows for exploring different trade-offs between financial performance and strategic alignment, and to identify the best solution for each target return level. The implementation of the case study, as well as the experimental setup, results, and validation, is available at https://github.com/weOptimize/softOpt_illustrative (accessed on 3 July 2026).

6. Computational Experiment

The computational experiment is designed as a methodological validation of the proposed GA-LLM soft-optimization framework rather than as a contribution to genetic algorithms for portfolio optimization. GA-based approaches for classical mean-variance portfolio optimization are already well established. In this paper, the GA is used as a hard-optimization module, based on a specific implementation developed within an ongoing project application, whose purpose is to generate a feasible shortlist of financially meaningful candidate portfolios. The main experimental objective is therefore to analyze the behavior of the LLM-based soft-evaluation layer and its interaction with the hard-optimization shortlist. In particular, the experiment assesses whether the LLM provides a distinct strategic signal from qualitative portfolio descriptors, and whether the trade-off parameter λ induces interpretable substitutions among candidate portfolios. The minimum-weight constraint is introduced to promote portfolio-composition diversity across hall-of-fame candidates; without it, shortlisted solutions at the same target-return level would tend to contain nearly identical asset sets, differing only through residual variations in portfolio weights.

The proposed methodology has been implemented in Python 3.11. The genetic algorithm was developed using the DEAP library, a widely used library for evolutionary computation. The algorithm was integrated with OpenAI's LLM through its API. All experiments were conducted on a standard personal computer equipped with 8 GB of RAM and an Intel Core i7 processor running at 2.3 GHz.

6.1. Reference Dataset

For this case study, stock market data that have been widely employed in previous studies is also used [31]. The dataset comprises constituents from five major stock market indices: Hang Seng (Hong Kong), DAX 100 (Germany), FTSE 100 (United Kingdom), S&P 100 (United States), and NIKKEI 225 (Japan). These data were obtained from DataStream and consist of weekly observations covering the period from March 1992 to September 1997. In particular, in this study, the Hang Seng (Hong Kong) dataset, referred to as port1, has been utilized. The benchmark instances were retrieved from the OR-Library repository and are publicly available at <http://people.brunel.ac.uk/~mastijb/jeb/orlib/> (accessed on 3 July 2026). This dataset is used as a standard benchmark in the scientific

literature to allow for comparison with existing best-known solutions. However, this dataset is extended and enriched with qualitative criteria to illustrate the capability of the proposed methodology. The aforementioned extension is done through a script, which assigns reproducible synthetic categorical labels to each asset from predefined pools (sector, country/geographic exposure, and ESG rating). These categories are appended as new dataframe columns, so the resulting dataset can be used downstream. It has been generated through a fixed pseudo-random seed for reproducibility reasons. Thus, each asset includes a token with the following structure:

```
asset_id | sector | country | ESG
```

For example:

```
13|Real Estate|Euro Area|A
```

This is the information that will be used by the LLM to evaluate how well a certain combination of assets fulfil the soft objective.

6.2. Hard Optimization Stage

The GA is executed across 200 generations with a population size of 50. It consists of the classic variance minimization problem associated with financial portfolios. It is executed a total of eight times, one for each target level. The target levels are calculated by taking the maximum possible return (100% of allocation to only one asset, the one with highest return), and dividing it by eight. For each target return, a HoF of the six best candidates is extracted. Therefore, there is a total of 48 candidates which constitute the HOS. Hence, each HOS candidate contains a stable candidate_id, the selected asset_ids, their respective weights, and their related asset tokens. These candidates are exported to the JSON format and used directly by the prompt-generation stage. The choice of JSON as an interface for LLMs stems from its simple, structured, and language-agnostic format, enabling reliable generation and straightforward parsing and validation by downstream systems.

6.3. Prompt Generation

The prompt generation uses a template-based architecture combining fixed instructional text with dynamically injected data. A predefined system message establishes the model's role as a financial portfolio evaluator and constrains its reasoning (e.g., prohibiting quantitative metrics), while a user template defines the task, embeds placeholders for asset info, rubric criteria, and candidate portfolios, and enforces a strict JSON output schema. The core pipeline ingests candidate portfolios with external files containing asset tokens and evaluation rubrics, then partitions candidates into batches to manage processing limits. For each batch, a function constructs a prompt payload by combining the template with batch-specific data and contextual inputs, enabling scalable and efficient processing of large candidate sets. Generated prompts are serialized into structured outputs (primarily JSON) for direct use in LLM inference, with optional plain-text versions for inspection. The design emphasizes modularity, reproducibility, and strict output control (via asset token verification), ensuring consistent structure, reliable parsing, and alignment with evaluation requirements.

For the purposes of strategic evaluation, solutions consisting of a single asset allocation are excluded from consideration. Specifically, only portfolios with a cardinality of at least two assets are retained. This restriction is imposed because meaningful portfolio analysis inherently relies on the interaction and diversification effects among multiple assets. Actually, a single-asset allocation does not constitute a portfolio in the strict sense and therefore does not allow for the assessment of strategic coherence, trade-offs, or alignment across complementary asset characteristics. A temperature of 0.2 is used to

ensure mostly deterministic outputs with slight randomness, improving consistency while avoiding overly rigid responses.

6.4. Rubric Implementation

The rubric is injected directly into the user prompt supplied to the LLM. The system message instructs the model to act as a financial portfolio manager, to evaluate only the rubric criteria, and not to use return, volatility, Sharpe ratio, efficient-frontier concepts, or other quantitative performance metrics. This separation is important: financial risk-return metrics are handled by the GA, whereas the LLM evaluates only soft strategic alignment. The current rubric contains four criteria, each scored on a 0–25 scale, as shown in Table 1.

Table 1. LLM Evaluation Rubric.

Criterion	Main Evaluation Focus
Sector Diversification & Balance	Balance across preferred sectors and avoidance of concentration
Geographic Structure	Developed-market core and controlled emerging-market exposure
ESG Quality	Consistency and strength of ESG ratings
Portfolio Coherence	Whether the selected exposures form a coherent portfolio

The LLM is asked to return one numerical rating per criterion and a final strategic S_{score} , defined as the sum of the criterion-level ratings. Thus, ESG quality, sector diversification, geographic structure, and overall coherence enter the soft-optimization layer through the rubric text itself rather than through additional hard-coded constraints.

6.5. Multiple Evaluations and Aggregation

To reduce sensitivity to single LLM calls, each prompt batch is evaluated multiple times. Candidate portfolios were divided into six batches of eight candidates each. Each batch was evaluated five times, yielding 30 LLM calls and score observations. To mitigate ordering bias, prompts were repeated with controlled variations: in every even-numbered evaluation, both batch order and candidate order within each prompt were reversed. The final strategic score used downstream is the mean of the repeated strategic_score values for each candidate, i.e.:

$$S_i = \text{mean}(S_{i1}, S_{i2}, \dots, S_{ir}) \quad (12)$$

where S_i is the aggregated strategic score of candidate i , and r is the number of LLM repetitions. The final fitness values of all HOS candidates are normalized using min–max scaling and computed according to Equations (10) and (11). Then, these candidates are evaluated across different values of λ (0, 0.01, 0.05, 0.1, 0.2, 0.5, and 1). For each target return, the selected portfolio may vary depending on the resulting fitness values. The parameter settings used to execute the GA are summarized in Table 2. These parameters were selected after a preliminary tuning phase. Instead of conducting an exhaustive parameter tuning study, we selected values that consistently provided stable convergence, adequate solution diversity, and reasonable computational times across all target-return levels.

Table 2. Baseline computational setup.

Item	Value
GA generations	200
GA population size	50
Tournament size	3
Crossover probability	0.90

Table 2. Cont.

Item	Value
SBX distribution index (η)	20
Mutation probability	0.2
Mutation distribution index (η)	20
Hall-of-Fame size	6
Target-return levels	8
HOS candidates	48
Prompt batches	6
LLM repeats per batch	5
LLM call temperature	0.2
Lambda values	0, 0.01, 0.05, 0.1, 0.2, 0.5, 1

7. Results

This section describes the results obtained in our experiments. First, it analyzes the hard optimization frontier. Secondly, it provides the LLM strategic scores. Finally, it performs an analysis of the λ parameter.

7.1. Hard Optimization Frontier

To assess the quality of the solutions generated by the GA, the resulting efficient frontier is compared with the unconstrained efficient frontier (UEF) available in the OR-Library for eight target-return levels. Thus, the frontier was divided into equidistant points along the vertical axis, which represents the user-defined expected portfolio return. Although the algorithm was designed to address the constrained portfolio optimization problem, this comparison provides insights about the effectiveness of the proposed approach and its ability to generate solutions that closely approximate the benchmark efficient frontier. The results show that the portfolio corresponding to the highest target return matches the benchmark solution exactly, as both select the single asset with the highest expected return. The largest deviation from the benchmark is observed at an intermediate target-return level. Nevertheless, the differences remain moderate and are consistent with the impact of the minimum-weight constraint. The resulting efficient frontier is shown in Figure 3. For each of the eight target returns, the minimum-variance point on the optimal frontier is marked with a cross ($\times$). The next five members of the hall of fame at the same target-return level are shown as circles ($\circ$) in the same color.

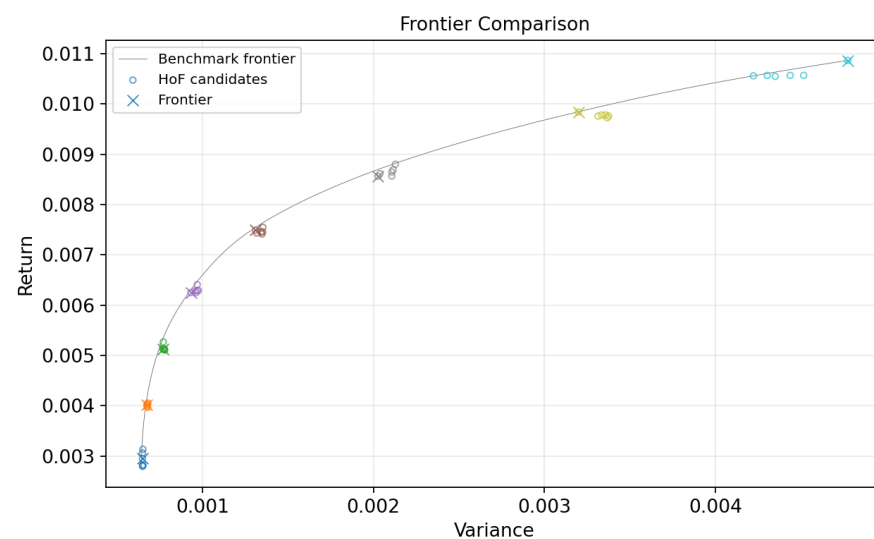


Figure 3. Hard optimization result with a 5% minimum weight constraint.

7.2. LLM Strategic Scores

The LLM generated a total of 240 raw scores across the 48 candidate portfolios. These scores ranged from 5 to 50, with a mean value of 19.82. The distribution of the raw scores is shown in Figure 4, where the black dots represent the individual sample S-scores obtained for each candidate. After aggregating the repeated evaluations, the mean candidate-level scores ranged from 9.00 to 27.60. The average standard deviation across repetitions was 5.40, indicating a non-negligible degree of variability in the LLM assessments. This result justifies the use of multiple evaluations and score aggregation, rather than relying on a single LLM call for each candidate portfolio.

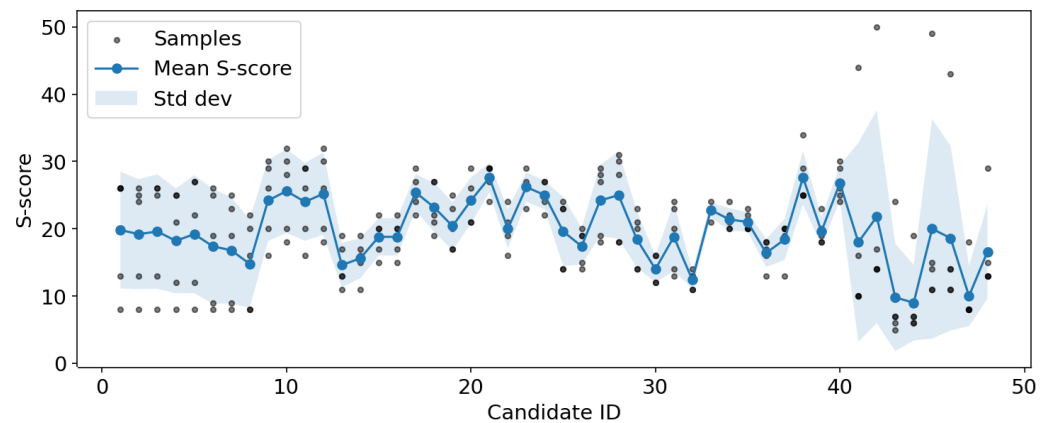


Figure 4. Strategic Scores provided to the HOS by the LLM after 5 runs.

In addition, the results highlight that high-scoring candidates are not necessarily those exhibiting the lowest variance within a given target-return group. Instead, they tend to be portfolios that better align with qualitative criteria such as sector balance, ESG characteristics, and overall diversification. This confirms that the LLM is not just detecting low-risk portfolios but is actively evaluating semantic alignment (e.g., ESG, sector balance, etc.), which is independent of financial risk metrics. Conversely, low-scoring candidates are typically more concentrated portfolios that prioritize high returns at the expense of diversification. These portfolios also exhibit greater variability in the LLM evaluations, suggesting a lower degree of consensus regarding their qualitative attractiveness. These findings open an interesting direction for future research, particularly regarding the stability of semantic evaluations across different LLMs, prompting strategies, temperature settings, and evaluation protocols.

7.3. Parametric Lambda Study

Figure 5 illustrates how the selected efficient frontier evolves as increasing weight is assigned to the soft objective, defined as $1 - S/100$. Each point is labeled as C_i, S_j , where C_i is the global identifier of the candidate portfolio in the HOS, while S_j represents its aggregated strategic score S derived from the LLM evaluation. Notice that the identifier T_k corresponds to each target-return level, and higher values ($k \in \{1, \dots, 8\}$) mean better returns.

At $\lambda = 0$, the selected portfolios correspond to the purely variance-driven solution. As λ increases, some points on the frontier are replaced by different candidates, as highlighted in the transition box in the figure. Each transition indicates that, for a given target-return level, the previously selected portfolio is no longer preferred once more weight is assigned to the strategic score. Instead, another candidate from the same target group (e.g., $C_7 \rightarrow C_9$ at T_2) is chosen because it provides a better balance between financial risk and strategic alignment. These changes do not imply large differences in variance, since the replacing portfolios are close in risk, but they reflect meaningful improvements in the soft (strategic)

objective. As λ continues to increase, fewer replacements occur, and for $\lambda \geq 0.5$ the frontier remains unchanged, indicating that the selection has stabilized and no further trade-offs are available within the candidate set. This indicates that beyond $\lambda = 0.5$, the cost of increasing strategic alignment outweighs the benefit in this specific candidate set, or that the candidate pool lacks sufficient diversity to further improve alignment.

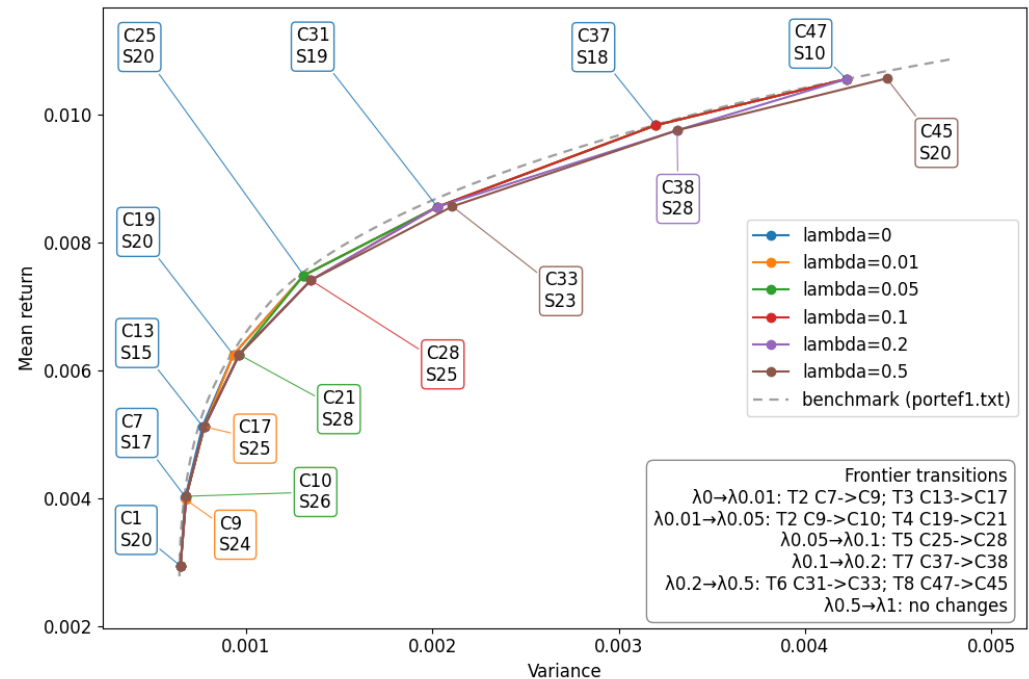


Figure 5. Efficient frontiers obtained for different values of λ .

The evolution of the portfolio quality metrics with different Lambda values can be seen both at Table 3 and Figure 6. Overall, increasing λ leads to a clear improvement in the average strategic score, which rises steadily from approximately 17.3 at $\lambda = 0.00$ to about 24.2 for $\lambda \geq 0.50$, where it stabilizes. This indicates that higher values of λ place more emphasis on the strategic component, resulting in portfolios with stronger strategic characteristics. In contrast, the average variance exhibits a slight but consistent increase as λ grows, moving from around 0.00172 to approximately 0.00178. Although the change is modest, this suggests a marginal increase in risk associated with prioritizing strategic objectives. Finally, the average return remains relatively stable across all values of λ , fluctuating within a very narrow range around 0.00684. A small decreasing tendency can be observed for higher λ , indicating that the gain in strategic score is achieved with only a negligible reduction in return. Taken together, these results highlight a trade-off where increasing λ enhances the strategic quality of the portfolio at the cost of a slight increase in risk and a marginal decrease in returns, with the effects plateauing beyond $\lambda \approx 0.50$.

Table 3. Evolution of quality metrics of the frontier portfolios across different lambda values.

λ	Frontier Points	Avg. S-Score	Avg. Variance	Avg. Return	Selected Candidates
0.00	8	17.300	0.001723	0.006851	1, 7, 13, 19, 25, 31, 37, 47
0.01	8	19.575	0.001724	0.006845	1, 9, 17, 19, 25, 31, 37, 47
0.05	8	20.650	0.001728	0.006851	1, 10, 17, 21, 25, 31, 37, 47
0.10	8	21.325	0.001733	0.006842	1, 10, 17, 21, 28, 31, 37, 47
0.20	8	22.475	0.001747	0.006832	1, 10, 17, 21, 28, 31, 38, 47
0.50	8	24.225	0.001784	0.006834	1, 10, 17, 21, 28, 33, 38, 45
1.00	8	24.225	0.001784	0.006834	1, 10, 17, 21, 28, 33, 38, 45



Figure 6. Evolution of quality metrics of the frontier portfolios across different lambda values. Each panel reports the evolution of one quality metric (S-score, variance and return).

8. Verification and Statistical Analysis

In this section we include the verification of the consistency of LLM outputs, as well as the significance of the strategic improvement across increasing values of λ .

8.1. Verification

A validation step is performed to ensure that the LLM outputs remain consistent with the original asset data. The LLM responses include structured asset tokens, which represent the asset features used during evaluation, i.e., its sector, geographic region and ESG rating. For the baseline experiment, a total of 1210 token mentions are recorded across the 30 LLM calls. All 1210 tokens match valid asset attributes, and no missing or inconsistent tokens are found. This yields a validation coverage of 100%. Notice, however, that while token consistency ensures accuracy of the input data, it does not guarantee the correctness of the LLM's semantic interpretation. Figure 7 shows, for each candidate, the graphical verification summary. In the horizontal axis the prompt identifier is depicted (b01/r01 means batch 1, repeat 1). The vertical axis shows the amount of tokens included in each prompt (in blue) and how many were OK (in orange). The green check mark (✓) indicates that all emitted tokens were classified as OK.

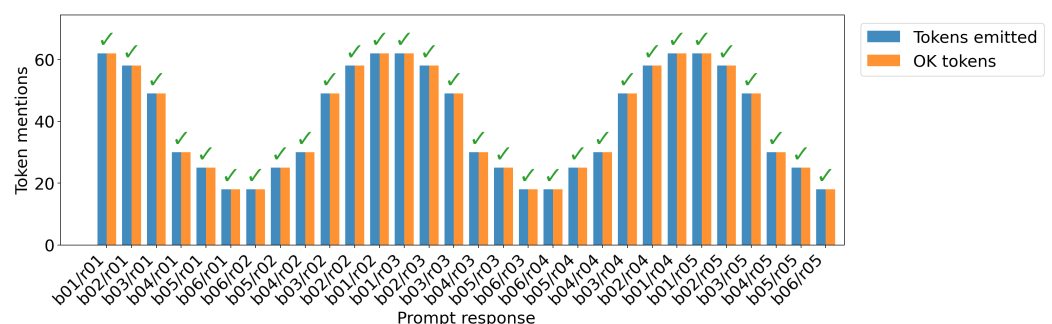


Figure 7. Summary of asset tokens emitted vs OK for each prompt.

These results indicate that the LLM preserves the identity and characteristics of the assets during evaluation, and that the strategic scores can be reliably linked to the underlying

ing candidate portfolios. Finally, the observed variability across repeated LLM evaluations, combined with full token consistency, suggests that the main source of variation is inherent stochasticity in the scoring process rather than data inconsistencies. This justifies the use of averaging in the final aggregation step.

8.2. Statistical Analysis

To confirm that the substitutions induced by the trade-off parameter λ correspond to systematic gains in strategic alignment (and to verify that the effect persists at a larger problem size), we performed a paired Wilcoxon signed-rank test on the aggregated LLM strategic scores $\bar{S}_i$ of the replaced and replacing candidates at each target-return level. Two instances were considered: (A) the illustrative 31-asset portfolio portef1 (Section 5), and (B) the larger 85-asset instance portef2 used in the scalability review (Section 9). Target-return levels whose selected candidate remained invariant across the parametric λ analysis were excluded from the paired analysis. Table 4 summarizes the eleven paired observations and the resulting statistics. In instance A, all seven substitutions yield strictly positive score gains ($\bar{\Delta S} = 8.0$, $\widetilde{\Delta S} = 9$), producing $W^+ = 28$, $W^- = 0$ and an exact two-sided $p = 0.0156$. In instance B, the four substitutions are also strictly positive ($\bar{\Delta S} = 8.25$, $\widetilde{\Delta S} = 3$), with $W^+ = 10$, $W^- = 0$ and $p = 0.125$; the direction of the effect is preserved, but the small sample precludes significance in isolation. Pooling both instances ($n = 11$) gives $W^+ = 66$, $W^- = 0$ and an exact two-sided $p = 2/2^{11} \approx 9.8 \times 10^{-4}$, rejecting the null hypothesis at $\alpha = 0.001$. The observed improvements are therefore not attributable to stochastic fluctuations of the LLM scoring and persist as the instance size grows from 31 to 85 assets.

Table 4. Paired substitutions used in the Wilcoxon signed-rank test. $\bar{S}$ is the aggregated LLM strategic score. Pairs are drawn from the λ -parametric analysis of each instance.

Instance	T	Replaced	$\bar{S}_{\text{old}}$	Replacing	$\bar{S}_{\text{new}}$	$\Delta \bar{S}$
portef1 (31)	T_2	C_7	17	C_{10}	26	+9
portef1 (31)	T_3	C_{13}	15	C_{17}	25	+10
portef1 (31)	T_4	C_{19}	20	C_{21}	28	+8
portef1 (31)	T_5	C_{25}	20	C_{28}	25	+5
portef1 (31)	T_6	C_{31}	19	C_{33}	23	+4
portef1 (31)	T_7	C_{37}	18	C_{38}	28	+10
portef1 (31)	T_8	C_{47}	10	C_{45}	20	+10
portef2 (85)	T_2	C_7	12	C_9	38	+26
portef2 (85)	T_3	C_{13}	34	C_{16}	35	+1
portef2 (85)	T_6	C_{31}	38	C_{32}	41	+3
portef2 (85)	T_7	C_{37}	8	C_{42}	11	+3

Instance A: $n = 7$, $W^+ = 28$, $W^- = 0$, exact two-sided $p = 0.0156$.

Instance B: $n = 4$, $W^+ = 10$, $W^- = 0$, exact two-sided $p = 0.125$.

Pooled: $n = 11$, $W^+ = 66$, $W^- = 0$, exact two-sided $p \approx 9.8 \times 10^{-4}$.

8.3. Robustness to Unchanged Target-Return Levels

A potential concern with the analysis above is that only target-return levels where the λ -parametric analysis produced a substitution are included in the paired test. Levels where the incumbent candidate remained selected across the whole parametric analysis (T_1 in portef1; T_1, T_4, T_5, T_8 in portef2) yield $\Delta \bar{S} = 0$ by identity, because both members of the pair are the same candidate and no operator was applied. Following standard practice, these ties were dropped (Wilcoxon [32], “wilcox” zero policy).

To verify that this choice does not drive our conclusions, we repeated the test on the full sample of 16 target-return levels (eight per instance) using two alternative zero policies

(Pratt’s rank inclusion [33] and the “*zsplit*” policy that allocates half of each zero rank to W^+ and W^-), and an exact sign test on the non-null pairs. Table 5 reports the resulting two-sided p -values. All four procedures reject the null hypothesis of no systematic gain at $\alpha = 0.005$ on the pooled sample and at $\alpha = 0.05$ on the smaller instance in isolation, confirming that the conclusion is insensitive to the treatment of unchanged target-return levels.

Table 5. Sensitivity of the Wilcoxon two-sided p -value to the zero treatment policy. All eight target-return levels per instance are retained; unchanged levels contribute $\Delta\bar{S} = 0$.

Zero Policy	portef1 ($n = 8$)	portef2 ($n = 8$)	Pooled ($n = 16$)
Wilcox (drop zeros)	0.0156	0.1250	0.00098
Pratt (rank inclusion)	0.0136	0.0484	0.00142
<i>z-split</i>	0.0138	0.0650	0.00168
Sign test (exact binomial)	0.0156	0.1250	0.00098

9. Scalability and Cost Review of the Soft-Optimization Layer

This subsection summarizes the scalability of the proposed LLM-assisted soft optimization layer. The focus is on the additional cost introduced by the strategic LLM evaluation, since the genetic algorithm (GA) component follows a standard optimization workflow and is not the main methodological novelty of this study. Two portfolio instances were executed. Portfolio 1 contains 31 assets, while Portfolio 2 corresponds to the larger 85-asset case, detailed in the next Section 9. The measured execution times and estimated costs are reported in Table 6. The estimates use Azure OpenAI GPT-5.4 Mini pricing, with an input-token rate of \$0.75 per 1 M tokens and an output-token rate of \$4.50 per 1 M tokens. Local compute cost is approximated using an Azure-equivalent CPU cost of \$0.05/hour.

Table 6. Pipeline Execution Times and Estimated Costs.

Stage	Portfolio 1 (31 Assets)			Portfolio 2 (85 Assets)		
	Time (s)	% Time	Cost (\$)	Time (s)	% Time	Cost (\$)
GA Optimization	125	25.9%	0.0017	290	30.1%	0.0040
LLM Evaluation	350	72.6%	0.0770	668	69.2%	0.1063
H.O.S. Export	1	0.2%	0.0001	1	0.1%	0.0001
Prompt Generation	2	0.4%	0.0001	2	0.2%	0.0001
Statistics Generation	3	0.6%	0.0001	3	0.3%	0.0001
Parametric λ Study	1	0.2%	0.0001	1	0.1%	0.0001
Total	482	100%	\$0.0791	965	100%	\$0.1107

Table 7 presents a breakdown of costs by pipeline stage. The LLM evaluation is the dominant monetary component of the pipeline, accounting for 97.4% of total estimated cost in Portfolio 1 and 96.0% in Portfolio 2. The other local stages, including H.O.S. export, prompt generation, statistics generation, and the parametric lambda study, are negligible in both time and cost. The GA stage is reported for completeness, but its scalability is not central to the contribution of this paper.

Table 7. Cost Distribution by Pipeline Stage.

Stage	Portfolio 1		Portfolio 2	
	Cost (\$)	% Total	Cost (\$)	% Total
GA Optimization	0.0017	2.1%	0.0040	3.6%
LLM Evaluation	0.0770	97.4%	0.1063	96.0%
Other Stages	0.0004	0.5%	0.0004	0.4%
Total	\$0.0791	100%	\$0.1107	100%

The cost estimate is based on the measured prompt logs. Portfolio 1 generated 31,025 prompt-log tokens, while Portfolio 2 generated 42,839 prompt-log tokens. Because exact Azure output-token accounting was not available, structured JSON output was estimated at approximately 11,940 tokens for Portfolio 1 and 16,490 tokens for Portfolio 2. The LLM cost was computed as:

$$\text{LLM Cost} = \frac{T_{in}}{1,000,000} \cdot 0.75 + \frac{T_{out}}{1,000,000} \cdot 4.50, \quad (13)$$

where T_{in} and T_{out} denote input and output tokens, respectively. This gives an input cost of \$0.0233, an output cost of \$0.0537, and a total LLM cost of \$0.0770 for Portfolio 1. For Portfolio 2, the input cost is \$0.0321, the output cost is \$0.0742, and the total LLM cost is \$0.1063. The observed increase from Portfolio 1 to Portfolio 2 shows that the total runtime increased from 482 s to 965 s ($2.00\times$), while the LLM evaluation time increased from 350 s to 668 s ($1.91\times$). The prompt-log size increased from 31,025 to 42,839 tokens ($1.38\times$), the estimated LLM cost increased from \$0.0770 to \$0.1063 ($1.38\times$), and the total estimated cost increased from \$0.0791 to \$0.1107 ($1.40\times$). These results indicate that the added soft-optimization layer remains inexpensive at the tested scales. Runtime grows more strongly than monetary cost, while LLM cost grows approximately with prompt-log size. This behaviour is mainly due to batching: each LLM request evaluates several H.O.S. candidates while reusing the same asset universe, rubric, verified asset facts, and JSON-response instructions. Consequently, the fixed prompt overhead is amortized across multiple candidate evaluations.

Hence, the scalability results support the practical feasibility of the proposed LLM-assisted evaluation layer. The main bottleneck is wall-clock latency rather than API expenditure. Future implementations can further improve scalability through parallel LLM batch execution, prompt compression, candidate pre-screening, response caching, and tighter JSON output schemas. Exact Azure token-usage logs should be used in future runs to replace the estimated output-token component with invoice-level accounting. The results for the Portfolio 2 instance are also included (Figure 8) for illustrative purposes. They show a comparable level of enhancement of the strategic quality of the portfolio at the cost of a slight increase in risk and a marginal decrease in returns.

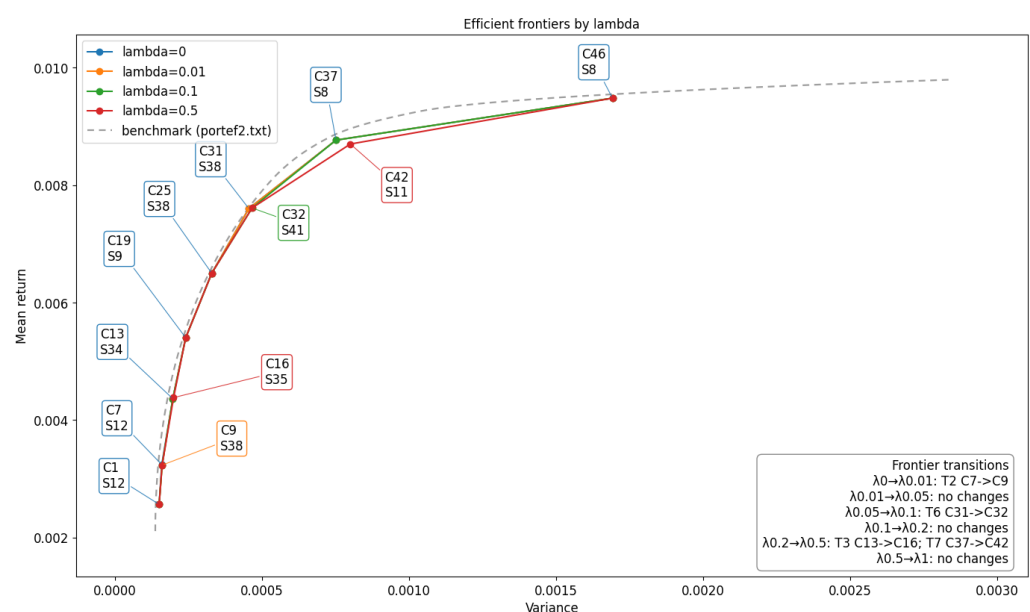


Figure 8. Efficient frontiers obtained for different values of λ for the instance Port2 (85 assets).

10. Managerial Insights

The proposed generative metaheuristics framework provides several practical implications for managers involved in complex decision-making environments where both quantitative and qualitative criteria are relevant. First, it enables more informed prioritization. Traditional decision tools often rely exclusively on numerical metrics such as cost, revenue, or risk. With the use of semantic evaluation through LLMs, managers can assess how well potential solutions align with strategic goals, organizational values, or stakeholder preferences. This allows for decisions that are not only financially sound but also consistent with long-term objectives such as sustainability, innovation, or social impact. Second, the hybrid approach supports flexibility and adaptability. In rapidly changing contexts, such as supply chain disruptions, market shifts, or policy changes, the framework allows new qualitative criteria to be incorporated without redesigning the optimization model. Managers can adjust prompts, evaluation criteria, or semantic objectives to reflect updated priorities, and the system can generate new candidate solutions accordingly.

Third, the framework encourages proactive scenario exploration. Employing LLMs to generate alternative configurations or policy-compliant solutions allows managers to evaluate options that may not have been considered in traditional numerical optimization. This promotes creative problem solving while maintaining adherence to hard constraints, leading to more robust and resilient decision-making. Finally, implementation considerations highlight trade-offs between computational cost and decision quality. Managers need to balance the frequency of LLM-based evaluations with time and budget constraints. The use of local or online LLMs, caching mechanisms, and surrogate models can help achieve efficient decision support without compromising the semantic alignment of solutions.

11. Conclusions

This paper introduced generative heuristics, a novel methodology that integrates LLMs with traditional metaheuristic algorithms to address multi-criteria optimization problems with distinct decision contexts where qualitative constraints require semantic interpretation. Combining the stochastic exploration capabilities of metaheuristics with the semantic reasoning power of LLMs allows the hybrid framework to simultaneously optimize quantitative financial metrics and qualitative strategic objectives. Empirical evidence from a portfolio optimization case study shows that this approach improves strategic alignment (e.g., ESG and sector balance) with low impact on financial risk compared to standard quantitative-only methods. Specifically, the genetic algorithm employed generated a hard optimization shortlist of 48 candidate portfolios across eight target-return levels. LLM-based semantic evaluations revealed a mean standard deviation of 5.40 across repeated assessments, highlighting the need to aggregate multiple queries to mitigate stochastic variability. Analysis of the weighting parameter λ indicated that strategic alignment significantly influenced portfolio selection only when $\lambda < 0.5$. Beyond this threshold, the efficient frontier stabilized, suggesting a saturation point where additional soft-objective weight yielded no further benefit. Furthermore, verification of processing integrity confirmed 100% token consistency across 1210 asset token mentions in 30 LLM calls, validating that semantic evaluations were accurate. Finally, results showed that high-scoring portfolios for strategic alignment were not necessarily the lowest-variance solutions but rather those with superior sector balance and diversification.

Despite these promising results, the study has certain limitations. First, the reliance on external LLM APIs introduces computational latency and cost, which may hinder scalability for real-time applications with large population sizes. Second, the quality of the semantic evaluation is inherently dependent on the pre-trained biases and capabilities of the specific LLM used. Hence, results may vary across different models or prompting strategies. Finally,

the current methodology assumes static qualitative criteria. Future work will focus on several directions to address these limitations. It is planned to investigate the use of smaller, fine-tuned local LLMs to reduce latency and enhance reproducibility. Additionally, another research line is to develop adaptive weighting mechanisms for the λ parameter, allowing the system to automatically balance hard and soft objectives based on search progress. Finally, extending the framework to handle dynamic, time-varying qualitative constraints and validating it on larger, more complex real-world datasets (e.g., supply chain or energy grid optimization) will further illustrate its practical utility.

Author Contributions: Conceptualization, A.A.J. and M.S.; methodology, A.A.J.; software, M.S. and J.P.; validation, J.P. and A.A.J.; formal analysis, J.P.; data curation, M.S.; writing—original draft preparation, M.S., A.A.J. and J.P.; writing—review and editing, A.A.J., M.S. and J.P.; supervision, A.A.J. and J.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially supported by the Spanish Ministry of Science, Innovation and Universities MICIU/AEI/10.13039/501100011033 (PID2022-138860NB-I00, AIA2025-163553-C44), the Barcelona City Council—La Caixa (2025 OVT 280226), and the Generalitat Valenciana (CIAICO/2024/117, CIBEST/2026/135).

Data Availability Statement: All data employed in the case study can be found at https://github.com/weOptimize/softOpt_illustrative (accessed on 3 July 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AEI	Agencia Estatal de Investigación
API	Application programming interface
CC BY	Creative Commons Attribution
CPU	Central processing unit
DAX	Deutscher Aktienindex
DEAP	Distributed Evolutionary Algorithms in Python
ESG	Environmental, social, and governance
FTSE	Financial Times Stock Exchange
GA	Genetic algorithm
GPT	Generative pre-trained transformer
GPU	Graphics processing unit
HoF	Hall of Fame
HOS	Hard optimization shortlist
JSON	JavaScript Object Notation
LLM	Large language model
MCDM	Multi-criteria decision-making
MICIU	Ministerio de Ciencia, Innovación y Universidades
NIKKEI	Nikkei Stock Average
NSGA-II	Non-dominated Sorting Genetic Algorithm II
OR-Library	Operations Research Library
RAM	Random-access memory
S&P	Standard & Poor's
SBX	Simulated binary crossover
UEF	Unconstrained efficient frontier

References

1. Martinsuo, M.; Killen, C.P. Value management in project portfolios: Identifying and assessing strategic value. *Proj. Manag. J.* **2014**, *45*, 56–70. [\[CrossRef\]](#)
2. Kock, A.; Heising, W.; Gemünden, H.G. How ideation portfolio management influences front-end success. *J. Prod. Innov. Manag.* **2015**, *32*, 539–555.
3. Brandenburg, M.; Rebs, T. Sustainable supply chain management: A modeling perspective. *Ann. Oper. Res.* **2015**, *229*, 213–252. [\[CrossRef\]](#)
4. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
5. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
6. Jiang, J.; Wang, F.; Shen, J.; Kim, S.; Kim, S. A survey on large language models for code generation. *ACM Trans. Softw. Eng. Methodol.* **2026**, *35*, 1–72. [\[CrossRef\]](#)
7. Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv. (CSUR)* **2003**, *35*, 268–308. [\[CrossRef\]](#)
8. Boussaïd, I.; Lepagnot, J.; Siarry, P. A survey on optimization metaheuristics. *Inf. Sci.* **2013**, *237*, 82–117. [\[CrossRef\]](#)
9. Markowitz, H.M. *Portfolio Selection: Efficient Diversification of Investments*; Yale University Press: London, UK, 2008.
10. Martí, R.; Pardalos, P.M.; Resende, M.G. *Handbook of Heuristics*; Springer Nature: Berlin/Heidelberg, Germany, 2025.
11. Valdez, F.; Melin, P.; Castillo, O. A survey on nature-inspired optimization algorithms with fuzzy logic for dynamic parameter adaptation. *Expert Syst. Appl.* **2014**, *41*, 6459–6466. [\[CrossRef\]](#)
12. Houssein, E.H.; Saeed, M.K.; Hu, G.; Al-Sayed, M.M. Metaheuristics for Solving Global and Engineering Optimization Problems: Review, Applications, Open Issues and Challenges. *Arch. Comput. Methods Eng.* **2024**, *31*, 4485–4519. [\[CrossRef\]](#)
13. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [\[CrossRef\]](#)
14. Cinelli, M.; Kadziński, M.; Gonzalez, M.; Słowiński, R. How to support the application of multiple criteria decision analysis? Let us start with a comprehensive taxonomy. *Omega* **2020**, *96*, 102261. [\[CrossRef\]](#)
15. Scott, J.; Ho, W.; Dey, P.K.; Talluri, S. A decision support system for supplier selection and order allocation in stochastic, multi-stakeholder and multi-criteria environments. *Int. J. Prod. Econ.* **2015**, *166*, 226–237. [\[CrossRef\]](#)
16. Ma, J.; Harstvedt, J.D.; Jaradat, R.; Smith, B. Sustainability driven multi-criteria project portfolio selection under uncertain decision-making environment. *Comput. Ind. Eng.* **2020**, *140*, 106236. [\[CrossRef\]](#)
17. Huang, S.; Yang, K.; Qi, S.; Wang, R. When large language model meets optimization. *Swarm Evol. Comput.* **2024**, *90*, 101663. [\[CrossRef\]](#)
18. Zheng, Y.; Zhang, L.; Li, K.; Wang, R.; Li, W.; Zhang, T.; Zhang, Q.; Jin, Y. A Survey on Large Language Models Driven Meta-Optimizers for Automated Intelligent Optimization. *Artif. Intell. Rev.* **2026**, *59*, 72. [\[CrossRef\]](#)
19. Li, M.; Luo, Q.; Xiao, Z.; Zhou, Y. Large Language Model Assisted Hyper-Heuristic Evolutionary Algorithm for Groundwater Level Prediction. *Sci. Rep.* **2026**, *16*, 21379. [\[CrossRef\]](#) [\[PubMed\]](#)
20. Ghanbarzadeh, R.; Ahadi Akhlaghi, I.; Ghafarian Gholamhossein, M.; Khan, M.N.; Mirjalili, S. Systematic Literature Review of Multi-Objective Hyper-Heuristics: A Human-in-the-Loop Large Language Model Methodology. *Artif. Intell. Rev.* **2026**, *59*, 131. [\[CrossRef\]](#)
21. Wu, X.; Wu, S.h.; Wu, J.; Feng, L.; Tan, K.C. Evolutionary computation in the era of large language model: Survey and roadmap. *IEEE Trans. Evol. Comput.* **2024**, *29*, 534–554. [\[CrossRef\]](#)
22. Du, S.; Zhao, J.; Shi, J.; Xie, Z.; Jiang, X.; Bai, Y.; He, L. A survey on the optimization of large language model-based agents. *ACM Comput. Surv.* **2026**, *58*, 1–37. [\[CrossRef\]](#)
23. Yu, X. Particle Swarm Optimization Meets Large Language Models. In *Proceedings of the 2025 IEEE 5th International Conference on Computer Communication and Artificial Intelligence (CCAI)*; IEEE: Piscataway, NJ, USA, 2025; pp. 1023–1028.
24. Da Ros, F.; Soprano, M.; Di Gaspero, L.; Roitero, K. Large language models for combinatorial optimization: A systematic review. *ACM Comput. Surv.* **2026**, *58*, 1–53. [\[CrossRef\]](#)
25. Long, S.; Tan, J.; Mao, B.; Tang, F.; Li, Y.; Zhao, M.; Kato, N. A survey on intelligent network operations and performance optimization based on large language models. *IEEE Commun. Surv. Tutor.* **2025**, *27*, 3915–3949. [\[CrossRef\]](#)
26. AhmadiTeshnizi, A.; Gao, W.; Udell, M. OptiMUS: Scalable Optimization Modeling with (MI)LP Solvers and Large Language Models. *arXiv* **2024**, arXiv:2402.10172. [\[CrossRef\]](#)
27. AhmadiTeshnizi, A.; Gao, W.; Brunborg, H.; Talaei, S.; Lawless, C.; Udell, M. OptiMUS-0.3: Using Large Language Models to Model and Solve Optimization Problems at Scale. *arXiv* **2026**, arXiv:2407.19633. [\[CrossRef\]](#)

28. Huang, C.; Tang, Z.; Hu, S.; Jiang, R.; Zheng, X.; Ge, D.; Wang, B.; Wang, Z. ORLM: A Customizable Framework in Training Large Models for Automated Optimization Modeling. *Oper. Res.* **2025**, *73*, 2986–3009. [[CrossRef](#)]
29. Wang, R.; Huang, Y.; Li, M.; Li, J.; Liang, D.; Simons, B.; Ke, P.; Liang, S.; Qin, K. Rethinking LLM-Driven Heuristic Design: Generating Efficient and Specialized Solvers via Dynamics-Aware Optimization. *arXiv* **2026**, arXiv:2601.20868. [[CrossRef](#)]
30. Fortin, F.A.; Rainville, F.M.D.; Gardner, M.A.; Parizeau, M.; Gagné, C. DEAP: Evolutionary algorithms made easy. *J. Mach. Learn. Res.* **2012**, *13*, 2171–2175.
31. Chang, T.J.; Meade, N.; Beasley, J.E.; Sharaiha, Y.M. Heuristics for cardinality constrained portfolio optimisation. *Comput. Oper. Res.* **2000**, *27*, 1271–1302. [[CrossRef](#)]
32. Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biom. Bull.* **1945**, *1*, 80–83. [[CrossRef](#)]
33. Pratt, J.W. Remarks on Zeros and Ties in the Wilcoxon Signed Rank Procedures. *J. Am. Stat. Assoc.* **1959**, *54*, 655–667. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.