Universiteit Leiden

Master Computer Science

LLM as scheduler: automatic design for HIIT via evolution

Name:	Johana Chen
Student ID:	s4159713
Date:	17/02/2026
Specialisation:	Artificial Intelligence
1st supervisor:	Dr. A.V. Kononova
2nd supervisor:	Dr. N. van Stein

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Acknowledgments

I would like to express my sincere gratitude to Dr. A.V. Kononova and Dr. N. van Stein for proposing such an interesting and engaging research topic and for their continuous guidance and support throughout this project. They dedicated significant time to meetings and discussions that were invaluable for the development and completion of this thesis.

I would also like to thank Robert Cabri for his valuable insights and constructive feedback, as well as for his assistance in connecting the project with professional coaches who contributed to the evaluation and refinement of the system.

My gratitude also goes to all participants who generously dedicated their time to take part in the experiments and made the effort to perform two HIIT programs.

Finally, I would like to thank my family for their continuous support, not only during this Master's program but throughout my entire academic journey. I am also deeply grateful to all my friends for their constant encouragement and emotional support. A special thanks to the friends I met during this Master's program, with whom I shared countless hours of study, deadline struggles, challenges, and growth. Their companionship and help made these two years truly memorable.

Last but not least, I thank God for His love and for guiding me throughout this amazing journey.

Abstract

Designing effective High-Intensity Interval Training (HIIT) programs typically requires professional expertise and is often carried out by experienced coaches. However, access to such expertise is often limited, creating a need for automated systems capable of generating structured, adaptable, and high-quality HIIT routines. In this thesis, we investigate the use of large language models (LLMs) within an evolutionary optimization framework to address this challenge. Inspired by the LLaMEA framework, we propose an LLM-based system in which candidate HIIT programs are generated, evaluated, and iteratively refined through an evolutionary loop.

A multi-component fitness function is introduced to quantify program quality, incorporating predicted physiological and biomechanical responses, structural characteristics, and user-defined constraints. The system explores four evaluation strategies: a fully fitness-driven approach, a human-in-the-loop configuration, a hybrid strategy combining automated evolution with human feedback, and a fully LLM-driven evaluation mode.

To assess real world applicability, we invited non-professional users to perform and evaluate two generated workouts, and asked professional coaches to interact with the system and assess it in terms of professionalism, accuracy, and overall quality. Their feedback suggests that LLM-assisted generation is particularly well-suited as a collaborative design tool.

This work demonstrates that combining LLMs with evolutionary mechanisms can be a potential tool for HIIT program generation. Addressing the limitations of previous related work, the proposed iterative optimization framework can systematically improve the quality of the outputs and prevent suboptimal solutions. Defining a fitness function can explicitly shape the program's quality. Furthermore, the comparison of different evaluation modes provides insight into the balance between human and LLM contributions within the optimization process.

Contents

Chapter 1

Introduction

1.1 Introduction to the topic

Designing an effective high-intensity interval training (HIIT) program is a complex task that often requires professional expertise and a deep understanding of training principles and physiological responses in humans. For instance, the work by Laursen et al. [30] explains how manipulating interval duration, intensity, and recovery fundamentally alters physiological stress and training adaptation. Therefore, a well-structured HIIT routine must carefully balance exercise selection, intensity, rest intervals, and overall workload in order to achieve specific training goals while avoiding overtraining or injury [19]. This complexity increases further when programs are intended to be versatile, long-term, and adaptable to different user preferences or fitness levels [10]. As a result, high-quality HIIT program design is typically reserved for experienced coaches and practitioners.

However, in everyday practice, access to professional coaching is often limited due to resource constraints, availability, or cost. At the same time, an increasing number of individuals are actively engaging in fitness and training activities, creating a growing demand for accessible tools that can support the design of effective workout routines without requiring professional level knowledge. This gap highlights the need for intelligent systems capable of assisting users in generating personalized and structured HIIT programs.

Recent advances in artificial intelligence have opened new possibilities for automating the creation of things and decision tasks [9][4]. In particular, Large Language Models (LLMs) have demonstrated strong capabilities in generating structured content, reasoning over constraints, and interacting with users by adapting outputs based on feedback expressed in natural language [37][19]. These characteristics make LLM a promising candidate for assisting or automating parts of the workout design process.

However, many current applications of LLMs rely on single-pass (one-shot) generation [11], where an output is produced in response to a prompt without mechanisms for systematic evaluation, refinement, and iterative optimization [49]. LLMs may also be influenced by biases in their training data [7], which can lead to suboptimal generations or premature convergence to locally optimal solutions. These limitations motivate the integration of LLMs with evolutionary computation, a family of optimization methods well-suited for iterative improvement and exploration of large design spaces [15]. In this thesis, we explore how LLMs can be embedded

within an evolutionary framework to act as intelligent schedulers for the automatic generation and optimization of HIIT programs.

1.2 Problem statement

Despite the expressive power of LLMs, generating high-quality HIIT programs still remains a challenging optimization problem. In this context, we classify a HIIT workout as high-quality if it satisfies several dimensions of training design. Firstly, the program must maintain an appropriate balance between exercise and recovery intervals, ensuring sufficient stimulus while preventing excessive fatigue accumulation. In addition, the intensity of the workout must reach a certain level able to improve cardiovascular fitness and muscular strength, yet remaining within safe physiological limits to reduce the risk of overtraining or injury. The selection of the exercises should also be adequately varied, either promoting balanced full-body engagement or aligning with a user-specific training focus. Finally, the progression and workload distribution must be coherent across intervals to maintain the training effective and sustainable.

A central difficulty lies in the formal evaluation of workout quality, as automated systems typically require a numerical fitness function capable of capturing multiple aspects of a workout routine.

Although recent LLM-based systems have demonstrated promising results in the automated generation of workout routines, they generally operate in a single-pass technique or rely on static prompting, without an explicit optimization loop or iterative refinement [2]. In contrast, traditional evolutionary algorithms require a fully specified objective function and struggle to incorporate high-level domain knowledge without extensive manual encoding.

Therefore, this thesis addresses the gap between these approaches by investigating how an LLM-driven evolutionary framework can be used to iteratively generate, evaluate, and refine HIIT programs. In particular, we focus on the challenge of designing a fitness function that is both expressive enough to reflect workout quality and efficient enough to guide the optimization process.

1.3 Research questions

Within this context, the central objective of this thesis is to design an LLM-based tool for evolving versatile HIIT programs adapted for long training periods. The study will focus on how HIIT program quality can be formally evaluated and optimized, and how different techniques of evaluation, automatic, human, or hybrid, affect the optimization process and its outcomes.

The main research questions guiding this thesis are:

RQ1. How to design a numerical formulation suitable for evolutionary optimization that can effectively represent the quality of a HIIT program?

RQ2. To what extent can an LLM be effectively integrated into an evolutionary framework for structured workout generation?

RQ3. To what extent is an evolutionary LLM-based HIIT generation system acceptable and useful to human users in practice?

1.4 Approach

To address these research questions, we propose an adaptation of the LLaMEA (Large Language Model Evolutionary Algorithm) [44] framework for the domain of HIIT program generation. In this system, the LLM is responsible for generating structured workout routines, while an evolutionary optimization loop guides iterative refinement based on fitness evaluations.

The core contribution of this work lies in the design of a multi-component fitness function that maps a HIIT routine to a numerical quality score. This function incorporates factors such as predicted physiological responses, workout intensity, work-to-rest ratios, and exercise variety.

In addition to a fully automated optimization process, we explore alternative evaluation and selection modes. These include a purely LLM-driven selection strategy, a human-in-the-loop setting, where users are in charge of the selection process and provide direct feedback on candidate programs, and a hybrid approach that combines automated evaluation with selective human input.

1.5 Thesis overview

After introducing the topic and outlining the research focus of this thesis, the overall structure of the document is presented as follows.

Before diving into details, Chapter 2 reviews relevant prior work, including studies on the use of LLMs as schedulers, the application of LLMs and evolutionary algorithms to program synthesis and optimization, the integration of LLMs within evolutionary optimization frameworks, and existing approaches to the automation of workout routine generation.

Chapter 3 then introduces the necessary preliminary concepts and presents the proposed system in detail. This chapter covers the workout numerical representation, the design of the fitness function, and the structure of the optimization loop.

Chapters 4 and 5 present the experimental setup and evaluation methodology, covering both automated and human-in-the-loop experiments, and discuss the results by analyzing observed trade-offs between different selection strategies.

Finally, Chapter 6 concludes the thesis with a detailed discussion of the proposed system, a reflection on the research questions, and outlines directions for future work.

Chapter 2

Related Work

The automatic generation of structured workout programs using LLMs is the result of combining different active research areas. These fields include task scheduling with language models, evolutionary optimization, prompt engineering, and personalized exercise recommendation systems.

Recent advances have shown that LLMs can act as high-level planners capable of organizing multi-step tasks under constraints [37], while evolutionary computation provides a principled framework for iteratively improving such structured outputs through mutation and selection. In parallel, LLM-driven evolutionary frameworks have emerged, using language models as operators within evolutionary algorithms to synthesize programs, strategies, or prompts [44]. Complementing this, work on automated prompt optimization highlights how the design of instructions and constraints influences the behavior and quality of LLM-generated content. Finally, research on automated workout routine generation and personalized training plan recommendation offers domain-specific foundations for representing and evaluating fitness routines.

The following sections review these research threads to contextualize the design choices and methodological contributions of our system.

2.1 LLM as scheduler

The use of LLMs has increased significantly in different applied areas, one of which is the application of LLMs as high-level schedulers. LLMs are often used to generate structured plans for specific objectives. This has led to a growing body of research at the intersection of automated planning and scheduling on the one hand, and language modelling on the other. Surveys of task planning with LLMs emphasize their ability to encode commonsense knowledge and to propose plausible multi-step strategies, while also highlighting limitations in reliability, constraint handling, and temporal reasoning. As discussed in recent work [50], LLMs are strong high-level planners as they excel at task decomposition, commonsense reasoning, and flexible goal interpretation. However, pure LLM planning lacks formal correctness, optimality, and safety guarantees. Therefore, the survey concludes that LLMs are powerful for generating and guiding plans, but robust task planning requires structured interaction with planning algorithms, tools, or environments, rather than relying on LLMs alone.

Often, LLM models plan by retrieving memorized patterns rather than actually reasoning

about actions and state changes. Valmeekam et al. introduce PlanBench [42], an extensible benchmark designed to evaluate the planning capabilities of LLMs beyond surface-level pattern completion. The benchmark is organized as a curriculum of tasks, including plan generation, plan verification, replanning, and robustness to goal reformulations. These tasks are grounded in classical planning domains such as Blocksworld and Logistics, with automated instance generation and correctness checking via planning tools. Experimental results on PlanBench and related benchmarks show that LLMs can generate plans that are syntactically valid and often qualitatively reasonable, but they struggle with longer horizons, strict precondition, effect logic, and precise constraint satisfaction. These limitations highlight the difficulty of relying solely on unconstrained prompting for complex planning tasks and motivate hybrid approaches that combine LLMs with explicit optimization, verification, or symbolic reasoning mechanisms. More recent studies further analyze the planning capabilities and failure modes of LLMs, confirming that purely generative approaches tend to exhibit unstable behaviour, especially when tasks involve non-trivial temporal or resource constraints. [31]

According to these findings, recent work has explored integrating LLMs into task planning systems while avoiding the assumption that LLMs can act as reliable standalone planners. Ahn et al. [1] propose grounding language in robotic affordances, using LLMs to select high-level intents while delegating low-level execution to trained controllers to drive how the movements should be performed. Similarly, Favier et al. [17] present an LLM-powered collaborative task planning framework in which LLMs support human–AI interaction by interpreting instructions, proposing high-level task decompositions, and enabling iterative refinement, while formal planning and execution are handled by downstream components. Both approaches emphasize a clear separation between language-based reasoning and structured decision making, demonstrating that robust task planning emerges when LLMs are embedded within systems that enforce constraints, feasibility, and execution correctness rather than relying on unconstrained language generation.

Overall, previous research suggests that LLMs are well-suited for high-level scheduling but require additional mechanisms to guarantee precision. Therefore, our framework is inspired to embed LLMs into a larger optimization loop to guarantee the quality of the results.

2.2 Program synthesis and optimization

Research has also focused on program synthesis, the automatic construction of programs or, more generally, the automated generation of structured procedures that satisfy a given specification or optimization objective [20]. The search spaces associated with such programs are typically large, discrete, and highly structured, making program synthesis a challenging problem. Consequently, candidate programs are often assessed through external evaluation signals, such as performance metrics, constraints, or user-defined preferences.

Direct solution generation often leads to suboptimal results, while manual program design and tuning are time-consuming and require significant domain expertise. Instead, automated program synthesis addresses these challenges by reducing human effort and enabling systematic improvement through iterative evaluation and feedback.

LLMs and evolutionary algorithms represent two complementary approaches to this problem. The following subsections review prior work on LLM-based and evolutionary approaches to

program synthesis and optimization.

2.2.1 LLMs for program synthesis

Recent advances in LLMs have led to renewed interest in program synthesis, particularly due to their ability to generate structured programs from natural language descriptions, examples, or partial specifications. LLMs trained on large corpora of code and structured text have demonstrated strong capabilities in synthesizing programs that are syntactically correct and semantically meaningful across a wide range of domains.

Much of the initial success of LLM-based program synthesis has been demonstrated in the context of source code generation. Systems such as DeepCoder [5] and later large-scale models like AlphaCode [32] showed that neural models can synthesize non-trivial programs by learning patterns from existing code repositories. However, these approaches largely rely on single-shot generation or beam search and do not explicitly optimize programs with respect to external performance criteria. As a result, the generated programs may be plausible but suboptimal, incorrect, or fragile when evaluated under strict constraints.

Parallel to these developments, research on symbolic and hybrid program synthesis has broadened the notion of a "program" beyond traditional source code. More recently, LLM-based systems have been used to synthesize robot control policies [1], task plans [17], and other structured procedures [33], further reinforcing the view that program synthesis encompasses any structured, executable representation of behavior.

2.2.2 Evolutionary algorithms for program synthesis

On the other hand, evolutionary algorithms have long been used for the automatic synthesis and optimization of programs, particularly in settings where the search space is large, discrete, and poorly suited to gradient-based optimization (e.g., methods that require continuous differentiable objective functions). In this context, program synthesis refers to the generation of structured artifacts, such as instruction sequences, that can be evaluated according to an external fitness function. The earliest and most influential research on automatic program synthesis using evolutionary methods comes from genetic programming. In genetic programming, a program is stored as a tree structure, representing the program itself. It adopts biological evolution operators (mutation and crossover) to generate new candidates and maximize task-specific objectives. Foundational studies by John R. Koza [28] introduced genetic programming, formalizing it as a general-purpose method for evolving programs. His work proved that this approach can automatically evolve programs and symbolic expressions directly from performance feedback (e.g., fitness score), without requiring labeled data or differentiable loss functions.

Beyond genetic programming, evolutionary strategies (ES) have also been widely adopted for program and policy optimization, emphasizing iterative refinement through mutation and selection. For instance, previous work has successfully applied $(1+1)$-ES to optimize symbolic controllers, rule-based systems, and algorithmic configurations [8]. It has been demonstrated to be robust to possible noise that recombination might bring, and ensures incremental changes through mutation.

2.3 LLMs in evolutionary computation

LLMs are increasingly being incorporated into evolutionary algorithms to enhance search and optimization processes. Previous work has explored the use of LLMs as mutation, recombination, or selection mechanisms within evolutionary frameworks.

One of the most common strategies for combining LLMs with evolutionary algorithms is to use LLMs as generative variation operators, such as mutation or crossover. LLMs are often applied to produce offspring solutions by rewriting or combining existing individuals in a semantically meaningful way. For instance, Evolution of Heuristics (EoH) [18] explicitly employs an LLM to perform crossover and mutation on candidate algorithms represented in natural language and executable code. Experimental results demonstrate that this approach is effective for automatic heuristic design, outperforming human-designed heuristics and other LLM-based code evolution methods such as FunSearch [38].

Similar approaches have also been explored, where LLMs guide evolutionary search by proposing structured modifications to candidate solutions, generating new heuristics, or providing reflective feedback during evolution. A recent survey by Chauhan et al. [12] gathers existing approaches that combine LLMs with evolutionary computation and identifies several common integration ways in which LLMs are used to improve evolutionary search. These include applying LLMs to generate candidate heuristics, using LLM-generated feedback to guide mutation over successive generations, employing LLMs as surrogate models to approximate expensive fitness evaluations from historical data, and extracting structural patterns from language models to guide the search towards more promising regions of the solution space.

However, most of the existing LLM-driven evolutionary approaches are largely empirical and rely heavily on heuristic design choices. In particular, the generation of new solutions is typically driven by prompt-based mutation or crossover operations, offering limited theoretical guarantees regarding convergence, robustness, or diversity preservation, while still requiring expensive fitness evaluations.

To overcome these limitations, recent work such as LLaMEA [44] explicitly frames LLMs as components within a principled evolutionary optimization loop. In this setting, the LLM is in charge of generating or refining candidate solutions, while evolutionary selection and explicit fitness evaluation determine the surviving candidate. This design aims to balance the expressive generative capabilities of LLMs with the robustness and stability of classical evolutionary computation.

2.4 Prompt optimization and automated prompt engineering

Prompting is an efficient method for adapting LLMs to specific tasks without updating their parameters through fine-tuning. However, one of the biggest limitations when using LLMs is that the model's performance is heavily influenced by the choice of the prompt. To address this issue, recent research has increasingly focused on the automation of prompt optimization.

Early approaches relied on gradient-based methods that require access to token probabilities or internal gradients of the LLM, such as prefix tuning and related techniques [40] [14]. Other methods emphasized exploration, using LLMs to generate large sets of diverse candidate

prompts and selecting the best-performing ones [51] [24], or exploitation-driven strategies, where an initial prompt is iteratively refined based on error analysis or feedback [22][36]. While effective in certain settings, these approaches have notable limitations: gradient-based methods are generally incompatible with API-based black-box models, exploration-focused strategies may waste resources evaluating low-quality prompts, and exploitation-focused methods risk premature convergence to local optima.

Therefore, several researchers focused on exploring the use of evolutionary algorithms in combination with LLMs to automate this prompt optimization process. EvoPrompt [21] is a representative framework that treats prompts as individuals within an evolutionary population and iteratively evolves them through LLM to generate new optimized prompt variants. Because prompts are discrete natural language objects, applying standard mutation and crossover on raw text might produce incoherent candidates. EvoPromot, instead, uses LLM itself as a variation operator, generating new prompts by combining parts from parent prompts and introducing controlled variations, such as rephrasing or word substitution, while preserving coherence. Candidate prompts are then evaluated on a development set using task-specific metrics, retaining only the top-performing prompts in the population. This approach has shown impressive results outperforming human-engineered prompts and other prior automatic methods, demonstrating that LLMs can effectively implement evolutionary algorithm operations for discrete prompt optimization and easily generalize across different tasks.

Beyond EvoPrompt, other systems also incorporated evolutionary elements into prompt tuning. The Automated Prompt Engineer (APE) [51], for example, integrates the LLM in both the prompt generation and evaluation loop, using heuristics or feedback based on the quality of the solution generated by the prompted LLM. For this approach, rather than maintaining a single best prompt, the authors introduced a Monte Carlo resampling step, in which new prompts are generated by mutating high-performing candidates, enabling a balance between exploration and exploitation during search.

2.5 Automated workout routine generation

The use of AI-driven approaches has also been explored in the field of personalized fitness, particularly for the generation and adaptation of workout routines.

Exercise is generally personal, and its adherence and effectiveness often depend on individual factors such as fitness level, heart rate response, and user preference. Therefore, machine learning models have been developed to personalize workout routines using physiological data and behavioral feedback. These systems aim to model real-time heart rate dynamics, optimize training intensity, and adjust training load over time through data-driven decision making. An influential study by Ni et al. introduced the FitRec model [35], a family of LSTM-based networks that uses multi-modal workout logs (heart rate, GPS, speed, etc.) to learn personal exercise patterns. The core idea of this approach is to move beyond simple statistics and build models that can truly understand an individual's unique fitness profile. The authors claim that their system could effectively estimate a user's heart rate profile for a given activity and recommend suitable workouts. However, the generalization of the system, data quality, and computational efficiency are still challenges. Subsequent work by Kayange et al. [26] proposed a more sophisticated model that combines the pattern recognition strength of FitRec's LSTM with a structured, physiological model based on Dynamic Bayesian Network (DBN). This

integration improved FitRec by building a more explainable and robust foundation, integrating domain knowledge with machine learning.

Beyond one-off predictions, reinforcement learning algorithms also enable continuous personalization of workout plans. Reinforcement learning systems treat exercise planning as an iterative decision process, updating routines based on user feedback and performance. For instance, *PERFECT* [2] was introduced as an exercise recommendation framework using a contextual bandit reinforcement learning algorithm. Instead of static plans, the authors proposed a dynamic, adaptive, and AI-powered personal trainer. These systems represent an important shift from static recommendations toward adaptive decision-making. However, they typically rely on predefined action spaces and structured representations of the exercises, which limit their flexibility in generating varied and personalized workout routines.

More recently, LLMs have been explored as an alternative paradigm for workout routine planning and generation. In contrast with task-specific algorithms, LLMs can leverage vast knowledge and conduct natural dialogues with users to generate highly diverse routines tailored to individual needs.

For example, PlanFitting [39] is an LLM-powered chatbot that helps users through conversation to create highly personalized weekly exercise plans. The agent dynamically asks about the user's goals, schedule constraints, and preferences, then generates a personalized workout plan aligned with established fitness guidelines. To ensure realism and adherence to expert recommendations, PlanFitting integrates Retrieval-Augmented Generation (RAG), grounding its suggestions in a database of exercises and professional rules. The authors claim that PlanFitting successfully guides users toward actionable plans and received positive feedback from physical therapist experts.

Khasentino et al. later introduced the Personal Health LLM (PH-LLM) [27], a fine-tuned model for fitness and sleep coaching. PH-LLM extends the Gemini LLM with multimodal adapters, enabling it to interpret both textual and numerical input from wearable devices, such as average heart rate or sleep duration variability. The system proved impressive performance, scoring higher than human experts on a fitness knowledge exam, and matched professional trainers in generating individualized exercise recommendations.

Although LLM-based approaches represent a significant step forward in user engagement and personalization, these systems mostly rely on single-pass generation without an explicit optimization process, and they lack formal mechanisms for evaluation and iterative refinement over time. These limitations motivate the exploration of the use of LLMs in combination with a more rigorous optimization framework. Such a hybrid approach sounds promising to enable deeper personalization and more robust adaptation than either component could achieve independently.

Chapter 3

Methodology

In this work, we apply an LLM evolutionary algorithm (LLaMEA) framework proposed in *LLaMEA: A Large Language Model Evolutionary Algorithm for Automatically Generating Metaheuristics* [44]. This consists of an approach that combines LLMs with evolutionary algorithms to generate optimized algorithms without requiring extensive prior expertise.

For the purpose of this thesis, the original LLaMEA framework is adapted to a fundamentally different application: the generation and refinement of HIIT programs. Instead of producing code, the system produces structured workout routines, and the evolutionary process optimizes these routines according to validity, safety constraints, physiological goals, and user or model-based preferences.

In this chapter, the methodological design of this adapted framework is described in detail, including the representation of HIIT programs, the mechanisms for candidate generation and mutation, and the selection and evaluation strategies that drive the evolutionary search. The four approaches (*LLMPredict*, *ABtest*, *Hybrid*, and *LLMchoose*) that define how candidate workouts are assessed and selected throughout the optimization process are also detailed. Each of them represents a distinct selection and evaluation strategy within the overall LLaMEA pipeline.

3.1 Preliminary knowledge

Before diving into the design of the system, it is necessary to present some background knowledge that will be assumed throughout the methodological descriptions. This section, therefore, provides an overview of evolutionary algorithms, a description of the LLaMEA framework, and a definition of HIIT programs.

3.1.1 Evolutionary algorithms

Evolutionary algorithms (EAs) [6] are optimization methods inspired by Darwinian evolution, which iteratively improve candidate solutions through a variation–selection cycle. The core idea of an EA is to maintain a population of candidate solutions, also referred to as individuals. Each individual is evaluated using a fitness score that measures its performance on a given task. The algorithm then repeatedly generates variant individuals, typically through mutation and/or

crossover, and selects the better-performing individuals to survive into the next generation. This process continues until a termination criterion is met.

The EA process follows a cycle composed of several key components:

1. **Initialization:** The process starts by creating an initial population of individuals. This step is typically performed randomly. However, in some cases, previously known high-quality solutions may also be used as a starting point.
2. **Evaluation:** Each individual in the population is then evaluated using a fitness or objective function. This step constitutes the core of the search process. The algorithm itself does not need to understand the problem being solved. Instead, it requires a mechanism to score candidate solutions by assigning a numerical value representing the quality or *fitness* of each solution for the given task.
3. **Variation operators:** To generate new candidate solutions, EAs typically employ two types of variation operators:
    - **Recombination/Crossover:** Crossover combines information from two or more parent individuals to create offspring. Common techniques include single-point, two-point, and uniform crossover. Crossover primarily supports exploitation by exploring promising regions of the search space.
    - **Mutation:** Mutation is a stochastic operator applied to a single individual, introducing small modifications to its genetic representation. Example techniques include bit-flip mutation for binary strings and Gaussian perturbations for real-valued vectors. The primary purpose of mutation is to promote exploration of the search space, prevent premature convergence to local optima, and maintain population diversity.
4. **Selection:** Selection guides the evolutionary process toward higher-quality solutions by favoring individuals with better fitness values. This mechanism is typically applied at two stages of the evolutionary cycle:
    - **Mating selection:** Also known as the parent selection, this step decides which individuals are chosen to create offspring. Fitter individuals generally have a higher probability of being selected as parents. Common methods include fitness-proportional selection or tournament selection.
    - **Survival selection:** This step determines which individuals remain in the population and survive to the next generation after the new offspring are created. This can involve selecting the best individuals from both the parent and offspring populations (a steady-state model) or replacing the entire parent population with the offspring (a generational model).
5. **Termination:** Finally, the evolutionary cycle is repeated until a termination criterion is satisfied. Common termination criteria include exhaustion of a computational budget, convergence of the population, achievement of a target fitness value, or a predefined maximum number of generations.

Figure 3.1 shows an illustrative representation of the EA cycle and the relationship between the main components.

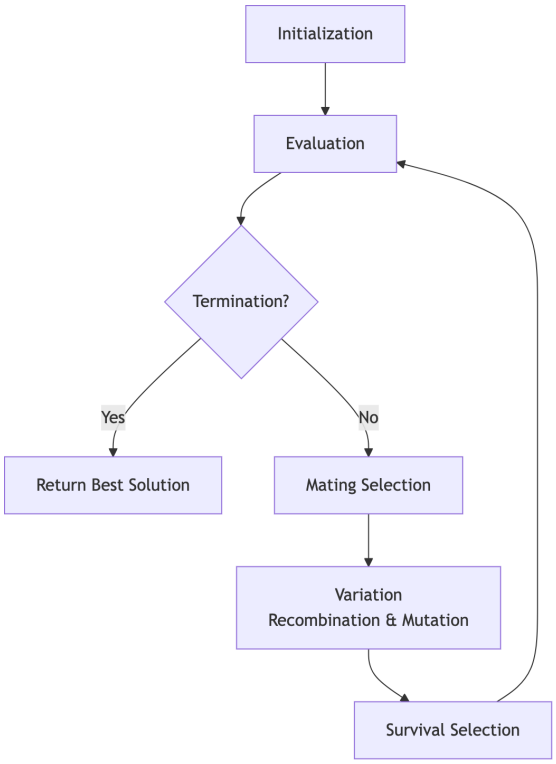

Figure 3.1: Schematic representation of the evolutionary algorithm (EA) cycle, illustrating the main components: initialization, evaluation, selection, recombination, mutation, and termination; and the overall iterative flow of the optimization process.

Depending on the specific methodology and representation used, evolutionary algorithms can be classified into several families. Four of the most common and well-established families are:

- **Genetic Algorithms (GA):** Genetic Algorithms traditionally employ fixed-length representations, most commonly binary strings, to encode candidate solutions. They strongly emphasize recombination (crossover) as the primary search operator, combined with mutation as a secondary source of variation. Selection is often fitness-proportional or tournament-based, promoting the survival and recombination of high-performing individuals.
- **Genetic Programming (GP):** Genetic Programming is specially designed to evolve computer programs or symbolic expressions rather than numerical vectors. Solutions are typically represented as tree structures, allowing programs of variable size and shape to be evolved. GP is particularly well-suited for tasks such as symbolic regression, automatic program synthesis, and rule discovery.
- **Evolution Strategies (ES):** Evolution Strategies were originally developed for continuous optimization problems involving real-valued parameters. They are characterized by a strong focus on mutation-driven search and the use of self-adaptive mechanisms to adjust strategy parameters, such as mutation step sizes, during evolution. Selection is commonly implemented using $(\lambda + \mu)$ or (λ, μ) schemes, which control how parents and offspring compete for survival.

- **Evolutionary Programming (EP):** Evolutionary Programming initially focused on the evolution of finite-state machines and later extended to real-valued optimization problems. In contrast to genetic algorithms, EP typically relies exclusively on mutation as its variation operator and does not employ crossover. Selection is usually based on stochastic tournaments among individuals.

The optimization task in this thesis involves continuous, high-dimensional parameter spaces, where candidate solutions are represented by real-valued parameters rather than discrete or symbolic structures. In such settings, Evolution Strategies are potentially effective, as they are explicitly designed to operate directly in continuous domains through mutation-based search mechanisms.

In particular, the $(1+1)$-ES offers a simple yet powerful optimization framework. By maintaining a single-parent solution and generating one offspring per iteration, it enables efficient exploration while requiring minimal computational resources. Although population-based methods could increase diversity by evaluating multiple candidate solutions per generation, they would also substantially increase computational cost and LLM usage. In the context of this thesis, such costs are non-trivial as fitness evaluations involve interactions with an LLM. While $(1+1)$-ES does not explicitly maintain diversity across multiple candidates, its reduced computational demand offers a practical balance between exploration and efficiency. Furthermore, its elitist selection mechanism ensures monotonic improvement in solution quality, making it well-suited for iterative refinement tasks.

3.1.2 LLaMEA

As mentioned previously, the inspirational foundation for this thesis is LLaMEA [44]. LLaMEA is a novel framework designed to automate the generation of metaheuristic optimization algorithms by integrating LLMs into an evolutionary loop. The objective is to overcome the inefficiencies of manual or expert-crafted algorithm design by iteratively generating, evaluating, and refining algorithms in a closed-loop process.

Within the LLaMEA framework, the LLM serves as an evolutionary generator of candidate algorithms. Apart from this new integration of an LLM, the overall structure of LLaMEA follows a classical evolutionary algorithm structure. The evolutionary loop starts with a task-specific prompt and an example algorithm, after which the LLM synthesizes an initial algorithm in code form. This first candidate is then evaluated using a specific benchmark platform, IOHexperimenter with the BBOB test suite [23]. Based on the evaluation, a scalar performance score is computed using the area over the convergence curve (AOCC) measure [34]. The resulting feedback, which includes runtime errors and detailed scoring metrics, is provided back to the LLM for prompting either to refine (mutate) or redesign the algorithm to produce a new candidate. Finally, the selection strategy determines whether only improved solutions are retained ($(1+1)$-strategy) or the newly generated candidates are always accepted ($(1,1)$-strategy). This evolutionary process is repeated for a fixed number of iterations (T).

An overview of the LLaMEA system is illustrated in Figure 3.2. The figure highlights the complete optimization loop, from problem initialization and LLM-driven algorithm synthesis to evaluation, error handling, and statistical performance assessment.

The authors demonstrated that LLaMEA can effectively generate high-performing algorithms that rival and sometimes even surpass existing state-of-the-art techniques, such as CMA-ES

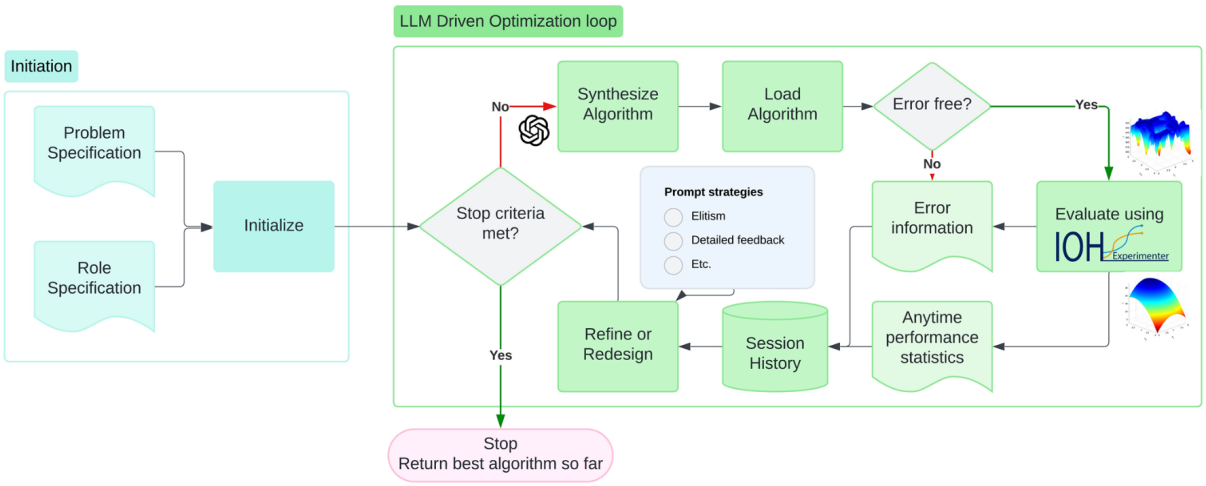

Figure 3.2: Schematic representation of the LLaMEA framework, depicting the evolutionary loop, including candidate generation via an LLM, fitness evaluation, selection, and iterative refinement, along with the information flow between components. Reproduced from [44].

[43] [46] or Differential Evolution (DE) [45]. Moreover, the framework highlights the potential of LLMs to innovate within the algorithmic design space, not only by fine-tuning algorithm parameters but also by introducing novel algorithmic logic and structural modifications.

While LLaMEA was originally proposed for the automated design of code-based algorithms, this thesis adapts its core principles to a different domain: the generation and optimization of HIIT workout routines expressed in natural language.

3.1.3 High Intensity Interval Training

HIIT is a training protocol alternating short periods of intense or explosive anaerobic exercises with brief recovery periods until the point of exhaustion [47]. Formally, HIIT is characterized by reaching at least around 90% of maximal oxygen uptake (VO_2max) or more than 75% of maximal power output during the work periods [3]. In practice, this means exercises are performed in repeated quick bursts at maximum or near maximal effort with periods of rest or low activity between bouts. Due to its extreme intensity, HIIT workouts are usually time-efficient and relatively short, under $20 - 30$ minutes for the entire session.

As introduced, a HIIT session is structured as a sequence of alternating work and recovery intervals. Therefore, besides the selection of exercises, the key programmable parameters include the work interval intensity and duration, the recovery interval intensity and duration, and the number of repetitions or cycles. A HIIT workout typically consists of a warm-up, multiple cycles of fast exercise followed by periods of slow recovery, and finally a cool-down. The work-rest ratios can widely vary, for example $1:1$, $2:1$, or $1:2$, depending on training goals. In general, shorter all-out intervals demand longer relative recoveries, whereas moderate-intensity intervals may use shorter or equal recovery durations. HIIT protocols are highly configurable, making them amenable to computational modeling. From an algorithmic perspective, a HIIT workout can be represented by a set of parameters such as interval durations, intensities, and exercise variability. Because these variables are explicitly defined, it becomes feasible to model

a HIIT program as a series of structured sequences that an AI or computational system could simulate or optimize for a given objective.

HIIT routines can take several forms and are often organized into named protocols. These protocols mainly differ in interval duration, number of repetitions, work-to-rest structure, and intensity. Some of the most commonly used formats are:

- **Tabata protocol:** The Tabata protocol was first introduced by Dr. Izumi Tabata [41]. It consists of 8 rounds of 20 seconds of very high-intensity exercise followed by 10 seconds of rest, traditionally performed at intensities approaching 170% of VO_2max. In practice, many contemporary Tabata-style workouts often adopt the same $20s$ work$/10s$ rest structure while operating at lower, more sustainable intensity levels. A key advantage of the Tabata format is its high degree of precision and programmability, as it is defined by fixed and well-structured timing intervals.
- **AMRAP (As Many Rounds/Repetitions As Possible):** As the name describes itself, in an AMRAP workout, the objective is to complete as many repetitions or rounds of a given exercise circuit as possible within a fixed time frame, typically around $10 - 20$ minutes [13][16]. Unlike interval-based protocols, AMRAP sessions do not prescribe explicit rest periods; instead, participants self-regulate rest breaks while the clock keeps running. This structure trains the body by promoting sustained effort and challenges both muscular endurance and cardiovascular capacity.
- **EMOM (Every Minute On the Minute):** EMOM workouts consist of performing a specific exercise or set of exercises at the start of each minute [13][16]. After completing the prescribed repetitions, the remaining time within the minute is used for rest. When the next minute begins, the cycle repeats. This structured timing creates a balance between high intensity and controlled recovery, making EMOM workouts both demanding and manageable.

In summary, HIIT programs are defined by a structured yet flexible combination of exercises, interval durations, intensities, and work-to-rest ratios. This balance between formal structure and configurability makes HIIT workouts particularly suitable for computational modeling and automated generation.

3.2 System framework

As introduced previously, the system developed is designed to automatically generate and iteratively improve HIIT programs by combining the generative capabilities of LLMs with the structured optimization process of an evolutionary algorithm. At a high level, the system's framework follows a classical evolutionary optimization loop: initialization, generation, mutation, evaluation, and selection. Each iteration of the loop refines the candidate HIIT programs toward workout routines that are valid, diverse, and aligned with human preferences or model-based fitness signals. In line with the original LLaMEA formulation [44], our implementation adopts a (1+1) evolutionary strategy, in which a single incumbent solution is maintained and challenged in each iteration by one newly generated or mutated candidate. The better of the two survives to the next generation, ensuring a simple yet effective form of steady-state optimization as only improvements are accepted. The overall framework of the proposed system

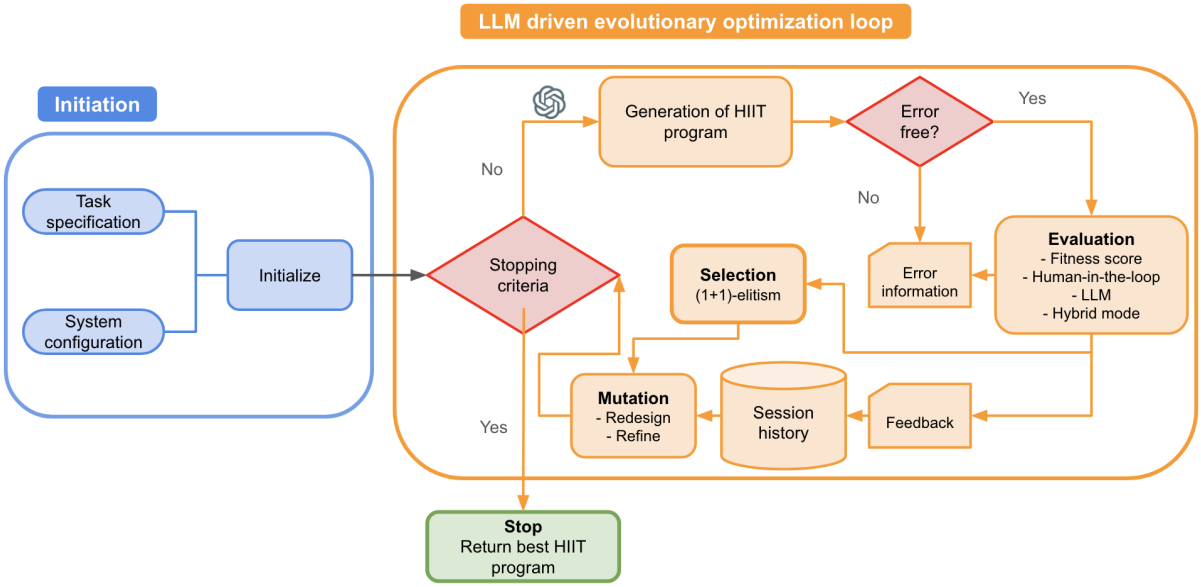

Figure 3.3: Schematic representation of the proposed system framework, depicting the evolutionary loop, including candidate generation via an LLM, fitness evaluation, selection, and iterative refinement, along with the information flow between components.

is illustrated in Figure 3.3.[1]

The optimization process begins with an initial HIIT program generated by an LLM according to a predefined schema. In each iteration, a new candidate is produced by the LLM and evaluated under one of the four selection and evaluation strategies: *LLMPredict*, *ABtest*, *Hybrid*, or *LLMchoose*. Each strategy provides different mechanisms for determining the candidate's quality, based on either model predictions, user preference, LLM comparison, or a combination of them. Finally, the candidate and current incumbent are compared, and the preferred one survives to the next generation. Iterations continue until a predefined computational budget is reached or a satisfactory workout program is found.

Overall, this framework is flexible, modular, and capable of incorporating different evaluation modalities, which allows it to explore the design space of HIIT workouts in a structured yet adaptive manner. In the following sections, we describe each component of this system in detail, including the representation of HIIT programs, the role of the LLM in generation and mutation, and the four evaluation strategies used for selection.

3.2.1 Pseudocode

Algorithm 1 describes the system's optimization loop used for structured HIIT program generation, which is an adaptation of the $(1+1)$-LLaMEA framework. The algorithm begins by generating an initial HIIT program using an LLM given the task prompt S (line 1). This initial candidate is then evaluated using the chosen evaluation function f, producing a fitness value and possible error information (line 2). The initial solution is stored as the best-so-far program (lines 3).

[1]The source code for this project is available at `https://github.com/JohanaChen/LLaMEA-for-HIIT/tree/Organized_code`.

Algorithm 1 (1+1)-LLaMEA for Structured HIIT Program Generation

Require: Task prompt S, feedback template F_0, budget T
Ensure: Best HIIT program P_b and its data y_b (including fitness)

1: $P_0 \leftarrow \text{LLM}(S)$ ▷ Initialize: generate the first HIIT program
2: $(y_0, e_0) \leftarrow f(P_0)$ ▷ Evaluate the program with evaluation function
3: $P_b \leftarrow P_0;\ y_b \leftarrow y_0$ ▷ Store the initial program as best-so-far
4: **while** $t < T$ **do** ▷ Loop until budget meeted
5: $F \leftarrow (S, \mathcal{H}, (P_b, y_b), M_t, F_0)$ ▷ Build new prompt: original task + previous program + current best + mutation + feedback
6: $P' \leftarrow \text{LLM}(F)$ ▷ Offspring generation with mutation
7: $(y', e') \leftarrow f(P')$ ▷ Evaluate the offspring program
8: **if** $e' \neq \emptyset$ **then** ▷ Errors ocurred
9: $y' \leftarrow -\infty$
10: **end if**
11: **if** $y' \geq y_b$ **then** ▷ (1+1)-selection
12: $P_b \leftarrow P';\ y_b \leftarrow y'$ ▷ Update best-so-far
13: **end if**
14: **end while**
15: **return** (P_b, y_b) ▷ Return best HIIT program and its data

The main optimization process is performed within the while-loop (lines 4–14), which iterates until the predefined budget T is exhausted or, in the specific case of the *ABtest* mode, the user enters the `STOP` command. At each iteration, a new prompt F is constructed (line 5) by combining the original task description, historical information H, the current best HIIT program and its fitness value, a mutation instruction specifying the type of variation to be applied M_t, and a feedback template F_0. The LLM is then queried to generate a new candidate HIIT program (line 6), which is subsequently evaluated (line 7). If runtime or structural errors are detected, the candidate is assigned the worst possible fitness value to penalize invalid solutions (lines 8–10).

Selection follows a $(1 + 1)$-elitist strategy, where the newly generated candidate replaces the current best solution if its fitness value is higher (lines 11–13). Once the budget is exhausted or the iteration is stopped, the algorithm returns the best HIIT program discovered during the optimization process with its corresponding fitness value (line 15).

3.3 Mutation strategy

In our proposed system, mutation is performed through prompt-based operators instead of traditional syntactic or parametric perturbations. Rather than directly modifying the HIIT programs, mutation is introduced by guiding the LLM through designed mutation prompts. This design choice aligns with the generative nature of LLMs and allows mutation to operate at a semantic level, affecting the structure, exercise selection, and pace of the HIIT programs.

At each generation step, the system probabilistically selects one of two mutation operators:

- **Refinement:** It consists of a local mutation operator, encouraging small yet incremental changes to the best-so-far HIIT program. These modifications may include slight adjust-

ments to exercise order, work-rest timing, or the replacement of at most one exercise with a similar alternative. Still, the overall structure, training format, and total duration of the workout are preserved.

- **Redesign:** Corresponds to a global mutation operator, allowing the system to explore substantially different regions of the search space by implementing drastic changes in the program. This includes significant changes in structure, style, intensity distribution, or exercise selection relative to previously generated programs. Redesign mutations are intentionally disruptive and are used to promote exploration, helping the system escape local optima and discover novel workout configurations.

The choice of these two mutation operators is governed by the mutation ratio parameter, which controls the probability of applying a more disruptive mutation (see Algorithm 2). This operator selection implements an explicit exploration-exploitation trade-off. While redesign operations induce larger semantic changes and therefore promote exploration of the search space, refinement operations perform local adjustments, encouraging exploitation of the current best solution.

Algorithm 2 Mutation operator selection function

Require: Mutation ratio $p_m \in [0, 1]$
Ensure: Mutation operator $op \in \{\textsc{Refine}, \textsc{Redesign}\}$
$r \leftarrow \text{Uniform}(0, 1)$
if $r < p_m$ **then**
    $op \leftarrow \textsc{Redesign}$ ▷ Exploration: generate a substantially different program
    $prompt \leftarrow redesign\_prompt$
else
    $op \leftarrow \textsc{Refine}$ ▷ Exploitation: apply small, incremental edits
    $prompt \leftarrow refine\_prompt$
end if
return $prompt$, op

Once a mutation operator is selected, a new prompt is constructed to generate the subsequent candidate solution. This prompt includes: the original task description defining the HIIT generation objective, the current best-so-far HIIT program, provided in a structured (JSON) format, the selected mutation instruction (refinement or redesign), and a specification of the required output format. In addition, depending on the selection and evaluation strategy, the prompt incorporates multiple forms of feedback when available. This includes system-generated feedback, such as evaluation results or fitness-related observations, user feedback from the most recent interaction, particularly relevant in *ABtest* or *Hybrid* modes, and accumulated feedback memory from previous rounds, which allows user preferences to persist across generations. The final prompt is then submitted to the LLM, which generates a new HIIT program that constitutes the mutated offspring and competes with the incumbent solution.

Overall, this prompt-based mutation strategy enables semantically meaningful variation while remaining fully compatible with the evolutionary optimization framework. It leverages the strengths of LLMs in structured generation and contextual reasoning, while preserving key evolutionary concepts such as mutation rate, operator diversity, and iterative refinement.

3.4 Selection and evaluation strategies

Another critical point that determines the effectiveness of the proposed system is how candidate HIIT programs are evaluated and selected for the subsequent generation. The selection methodology in our system aims to explore, apart from the classical fixed fitness function, different levels of human involvement. To this end, we designed four distinct selection and evaluation modes: *LLMPredict*, *ABtest*, *Hybrid*, and *LLMChoose*.

These modes span a spectrum from fully automated evaluation based on model predictions to human-in-the-loop selection strategies. The design of multiple selection modes allows the system to be flexible in different experimental contexts. In particular, it enables controlled comparisons between automated and human-driven evaluations.

This section provides detailed descriptions of each selection mode.

3.4.1 LLMPredict

In this mode, following the classical paradigm of evolutionary algorithms, candidate HIIT programs are evaluated using a scalar fitness function. Each generated program is represented as a numerical score that describes its overall quality, and selection is performed by choosing candidates with higher fitness values. Therefore, this mode consists of a fully automated evaluation mode without the intervention of human feedback.

Initial fitness formulation

The initial design of the fitness function aimed to capture the quality of the HIIT program through a weighted combination of domain-inspired criteria, including intensity, workout density, exercise selection, progression, and scalability. We could express the evaluation score as:

$$s = w_1 I + w_2 D + w_3 E + w_4 P + w_5 S,$$

where w_i are the weighting coefficients and:

- I denotes the intensity, estimated using Metabolic Equivalent of Tasks (METs),
- D represents the density, work-to-rest ratio,
- E evaluates the quality of the selected exercises,
- P captures progression and overload structure,
- S measures scalability and safety considerations,

While this formulation allowed the system to reason about structural and temporal properties of the workout routines, it exhibited one fundamental limitation: the components involved in the fitness score are mainly derived from the workout duration, intensity, or predefined heuristics. As a result, the fitness score lacks the direct expression of the physiological effect of performing the workout on the human body.

Incorporating predicted physiological and biomechanical responses

To overcome this limitation, we introduce an evaluation strategy that explicitly models the expected effects of an HIIT program rather than relying solely on its surface-level structure.

Inspired by studies on athletes' performance modeling, particularly the study by Qin et al. [25], athletic performance is understood to emerge from the intersection of physiological, biomechanical, psychological, and environmental factors.

According to the paper, the physiological domain encompasses the internal biological processes that support athletic performance, such as cardiovascular efficiency, metabolic capacity, and recovery dynamics. The biomechanical factors represent the fundamental movement patterns and physical capabilities that directly impact performance execution, for example, acceleration, power, agility, strength, and flexibility. The psychological domain encompasses the mental factors that influence an athlete's ability to optimize physical capabilities and maintain performance under pressure, such as mental toughness, athlete engagement, passion, group cohesion, or resilience. Finally, the contextual domain accounts for the environmental and situational factors that influence performance, including competition level, team versus individual sport dynamics, training environment characteristics, and seasonal phase.

Based on this insight and considering the available resources, we decided to integrate a physiological factor and a biomechanical factor into the fitness function. To do so, we leverage the LLM to predict human response variables to the generated HIIT programs. Therefore, in addition to extracting program-level properties such as work ratio, intensity, and exercise variety, the LLM is also prompted to estimate the **heart rate** response as a physiological factor and the **power output** as a biomechanical factor.

Shaped fitness function for optimization using Gaussian scoring

To make the evaluation compatible with the optimization evolutionary loop, several components of the fitness function are computed using a **Gaussian scoring function**. This technique assigns the highest score to values close to a predefined target, and gradually penalizes deviations according to a standard deviation parameter.

Formally, the Gaussian distribution or normal distribution is defined as [48]:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

where μ is the mean or expectation of the distribution and σ^2 is the variance. However, for our purpose, we employ a **Gaussian-shaped scoring function** to evaluate the fitness components. Unlike the Gaussian probability density function, the normalization constant is omitted, as the function is not used to model a probability distribution but rather to express preference for values close to a desired target. This formulation yields a bounded, smooth score in $(0, 1]$, suitable for fitness shaping in our evolutionary optimization setting. Therefore, the expression for each component becomes:

$$\text{score}(z) = \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right),$$

where μ represents the desired target value, and σ^2 the variance.

This approach allows the system to encode soft physiological constraints, such as preferring heart rate responses that are challenging yet safe (e.g., 70-90% of estimated maximum heart rate), dense but recoverable work ratios, and moderately high average intensity. In contrast to

hard thresholds, Gaussian scoring provides a smooth and interpretable way to guide optimization toward realistic and desirable regions of the search space.

This Gaussian mechanism is applied to all the components of the fitness score, except for the exercise variety, for which a monotonic saturation function is applied instead, encouraging diversity while avoiding unbounded growth in the score once sufficient variety is achieved.

Final fitness score

Finally, the fitness score is defined as:

$$s = w_1 \cdot \mathsf{score}(z_{hr}) + w_2 \cdot \mathsf{score}(z_{power}) + w_3 \cdot \mathsf{score}(z_{work}) + w_4 \cdot \mathsf{score}(z_{intensity}) + w_5 \cdot z_{variety}. \tag{3.1}$$

where $z_i \in [0, 1]$ denotes a normalized score derived from the corresponding predicted or extracted feature, score() depicts the Gaussian score function, and the coefficients $w_i \in [0, 1]$ denote the weights assigned to each component satisfying $\sum_i w_i = 1$.

This formulation enables a fully automated evaluation pipeline in which LLM-predicted physiological and biomechanical responses are integrated into a structured fitness function. With this design, we aim to bridge the gap between abstract workout descriptions and their expected human impact, while remaining compatible with evolutionary optimization principles.

3.4.2 ABtest

In the *ABtest* mode, the automated fitness function is designed to be entirely replaced by **human evaluation**, giving the user direct control over the selection process within the evolutionary loop. Instead of assigning a scalar fitness score to each candidate solution, the system relies on pairwise comparisons between HIIT programs.

At each iteration, the system presents two HIIT workouts: the current incumbent (parent) solution and a newly generated candidate (offspring) solution. Then, the user is asked to evaluate both workout routines and provide one of the following inputs:

- **A**: Choose the current incumbent if this one is preferred.
- **B**: Choose the new candidate program if it is preferred.
- **T**: Declare a tie if both routines are considered equivalent in quality, in which case the system randomly selects one of the two.
- **STOP**: Terminate the optimization process if the current incumbent is already considered satisfactory.

In addition to the selection decision, users may also provide textual feedback describing their preferences, perceived shortcomings, or any additional instructions to the system. This feedback is then incorporated into the prompt used for subsequent candidate generation, allowing the system to adapt not only through selection pressure but also through explicit qualitative guidance.

In contrast to the fully automated evaluation mode, this approach implements a human-in-the-loop evaluation process, making it particularly suitable for studying the comparison between automated metrics and subjective evaluation by users.

3.4.3 Hybrid

The Hybrid mode is designed as the result of combining the scalability of automated evaluation with the expressiveness of human judgment. In this setting, the system alternates between the fitness evaluation from *LLMPredict* and the human selection in *ABtest*.

Concretely, the evolutionary process proceeds for a predefined number of iterations using the *LLMPredict* fitness function, during which candidate programs are automatically evaluated and selected based on their scalar fitness scores. After every predefined x number of automated iterations, a human evaluation step is introduced. During this step, the user compares the current incumbent with a newly generated candidate using the same pairwise selection mechanism as in the *ABtest* mode.

This strategy aims to enable the system's optimization process to explore the search space efficiently using automated guidance, while periodically correcting or refining the search direction based on human preferences. Therefore, the Hybrid mode balances optimization efficiency with alignment to subjective user expectations.

3.4.4 LLMChoose

Finally, in the *LLMChoose* mode, the evaluation and selection process is completely delegated to the LLM. Similar to the *ABtest* mode, this approach does not rely on an explicit fitness function. Instead, selection is again performed through pairwise comparisons between candidate solutions. However, rather than relying on user preference, the LLM itself performs the qualitative comparison and selects its preferred solution under predefined guidance.

As in the *ABtest* mode, in each iteration, the LLM is presented with two HIIT programs: the current incumbent solution and a newly generated candidate. The model is prompted to compare both programs while adopting the role of a professional fitness trainer. The evaluation is guided by a fixed set of high-level criteria, including exercise variety, balance between training modalities, structural coherence, appropriate duration and rest intervals, and suitability for an intermediate fitness level. Based on this comparison, the LLM is instructed to select the preferred program and provide a brief justification for its decision.

Although explicit evaluation criteria are provided, no quantitative fitness function is imposed, positioning the *LLMChoose* mode as an intermediate approach between fully automated fitness-based evaluation and human-in-the-loop selection. This framework effectively explores whether an LLM can act as a coherent qualitative evaluator, applying domain knowledge and holistic judgment without explicit numerical optimization objectives.

Chapter 4

Experiment

4.1 Experiment 1: Comparison of LLMs

4.1.1 Objectives

To evaluate the influence of the underlying LLM on evolutionary performance, we conducted a comparative study across multiple LLM backends. For each model, the system was executed for 30 generations and repeated six times using different random seeds. During each run, the best-so-far fitness value was recorded at every generation. The resulting fitness trajectories were averaged per generation, and both individual runs and mean performance curves were visualized to assess convergence behavior, stability, and final solution quality.

4.1.2 Experiment setup

To isolate the effect of the LLM model on the system's optimization performance, we conducted a controlled comparison across four LLM backends: GPT-4o-mini (OpenAI), Gemini 3 Flash (Google), DeepSeek-Chat (DeepSeek), and Llama-3 (Ollama, local inference). To ensure a fair comparison, all features of the optimization process were kept identical across experiments, including task definition, prompting strategy, solution representation, and evolutionary hyperparameters.

The system was run for a fixed budget of 30 iterations, repeated 6 times for each configuration. Offspring were generated using a fixed mutation framework that alternates between refinement and redesign operators, with a mutation ratio of 0.2, meaning that redesign mutations were applied with a probability of 0.2 in each generation.

For Experiments 1–3, the evaluation mechanism is based exclusively on the proposed multi-component fitness function (*LLMPredict*). The weighting coefficients are fixed throughout these experiments to ensure comparability across configurations. Specifically, the weights are set as follows: $\{w_1 = 0.3, w_2 = 0.25, w_3 = 0.2, w_4 = 0.15, w_5 = 0.1\}$.

4.2 Experiment 2: Effect of mutation rate

4.2.1 Objectives

Mutation plays a crucial role in evolutionary algorithms as it balances the exploration-exploitation trade-off. In our system context, the mutation rate determines how often offspring are generated through substantial changes (redesign) or small improvements (refinement). While previous work in evolutionary computation has shown that mutation significantly affects convergence speed, stability, and solution quality [29], its impact in LLM-driven evolutionary systems remains underexplored.

Therefore, the objective of this experiment is to systematically explore how different mutation rates influence the optimization dynamics and final performance of our system in the generation of HIIT workouts.

4.2.2 Experiment setup

Based on the results of Experiment 1, GPT-4o-mini is selected as the LLM backend for all subsequent experiments, as it provided the best balance between convergence speed, consistency across runs, and final fitness performance. All features of the optimization process are kept identical to Experiment 1, except for the mutation rate.

In this experiment, we evaluate the following mutation rate settings:

$$\{0.0, 0.3, 0.7, 1.0\},$$

where the mutation rate denotes the probability of applying a redesign mutation in a given generation. The selection of mutation rates aims to span the full spectrum of exploration-exploitation trade-offs. Mutation rate 0.0 represents a purely exploitative approach, where only minimal refinement mutations are applied, allowing the system to perform local improvement. Mutation rate 0.3 introduces a small level of exploration, favoring refinement while occasionally introducing redesign mutation to escape from local optima. Instead, mutation rate 0.7 biases the search toward unexplored solution areas, enabling frequent major changes while still retaining some capacity for incremental improvement. Finally, the mutation rate 1.0 fully explores the search area, where every offspring is generated through redesign mutation. With this range, we aim to systematically analyze the effect of increasing mutation strength in evolutionary dynamics in the system.

Each configuration is executed for 30 generations and repeated 3 times. Similar to Experiment 1, the best-so-far fitness is recorded at every generation, and individual and mean performance curves are analyzed to assess convergence speed, stability, and final solution quality.

4.3 Experiment 3: Validity of LLM's predictions

Experiment 3 consists of the validation of the heart rate predictions generated by the LLM. These predictions are a core component of the fitness evaluation function used in our system; therefore, their reliability is crucial for the correctness of the optimization process.

To assess prediction accuracy, participants are provided with two generated HIIT workout programs. For each workout, participants are asked to record their heart rate (in bpm) at

multiple time points, including at the beginning of the workout routine, after each exercise and rest interval, and at the end of the session. The heart rate measurements are collected through the participant's own wearable device or fitness tracker.

Then, the recorded values are compared to the corresponding LLM-predicted heart rate trajectories. Predicted and observed values are visualized as individual and mean time-aligned curves, and analyzed to assess deviations, trends, and peak alignment. This comparison provides insight into whether the LLM is able to predict realistic heart rate dynamics during HIIT workouts.

4.4 Experiment 4: User feedback

At the end of the pipeline, the generated workout routines are designed for practical human use. Therefore, Experiment 4 focuses on user feedback and evaluation of the provided HIIT workouts in Experiment 3. The goal for this experiment is to explore how users perceive the quality, intensity, and overall suitability of the generated workouts.

Participants are invited to perform both HIIT routines, each at a separate time. After completing each workout, they are asked to fill in a structured questionnaire collecting both background information and subjective evaluations. The questionnaire first gathers demographic and activity-related information, including age, regular workout habits, and weekly training frequency. This is followed by numerical ratings on a $1-10$ scale assessing perceived power, physical demand, work-rest ratio, intensity, exercise variety, and overall workout quality. These evaluation dimensions are intentionally selected to match as closely as possible with the definition of the fitness function used by the system, to have a fairer comparison between human judgments and the system's internal evaluation criteria. Finally, participants are asked whether they would be willing to regularly use the HIIT routines.

After completing both workouts, participants are also asked to indicate which of the two HIIT programs they prefer overall. This final comparison provides a direct preference-based evaluation between both routines and allows for a comparison between the system's preference and the users' actual preferences.

4.5 Experiment 5: Trainer ABtest

This final experiment aims to investigate the usability and perceived value of the system from a professional coaching perspective. Instead of evaluating the final outputs of the system, this experiment focuses on how experienced fitness coaches interact with the system and whether, from a professional standpoint, it provides meaningful, actionable insights for workout design, and is worth developing further.

Coaches are invited to interact with the system in the A/B testing mode, which allows them to compare alternative HIIT program generations and provide feedback after each generation. Each coach uses the system at least once to explore how it responds to preference signals and iterative refinement.

After completing the interaction, coaches are asked to fill in a structured feedback form, where they need to use a $1-10$ scale to evaluate the professionalism of the generated HIIT routines,

the accuracy of the system's response to feedback, the diversity of the generated solutions, and the overall quality of the final output. Additionally, they are asked whether they would prefer to use such a system for generating workout routines or rely on manual program design.

Finally, open-ended feedback is also collected to identify any limitations of the system and to gather suggestions for potential improvements and future directions.

Chapter 5

Results

5.1 Experiment 1: Comparison of LLMs

As mentioned previously, the four language models are selected to represent a diverse range of architectures, deployment settings, and performance profiles. Specifically, the comparison includes a compact proprietary model optimized for fast inference (*gpt-4o-mini*), an open-source locally deployed model (*llama3*), and two alternative conversational models with different training and inference characteristics (*deepseek-chat* and *gemini-flash*). This selection enables an analysis of how model capability, inference cost, and response quality influence the behavior of the evolutionary optimization process.

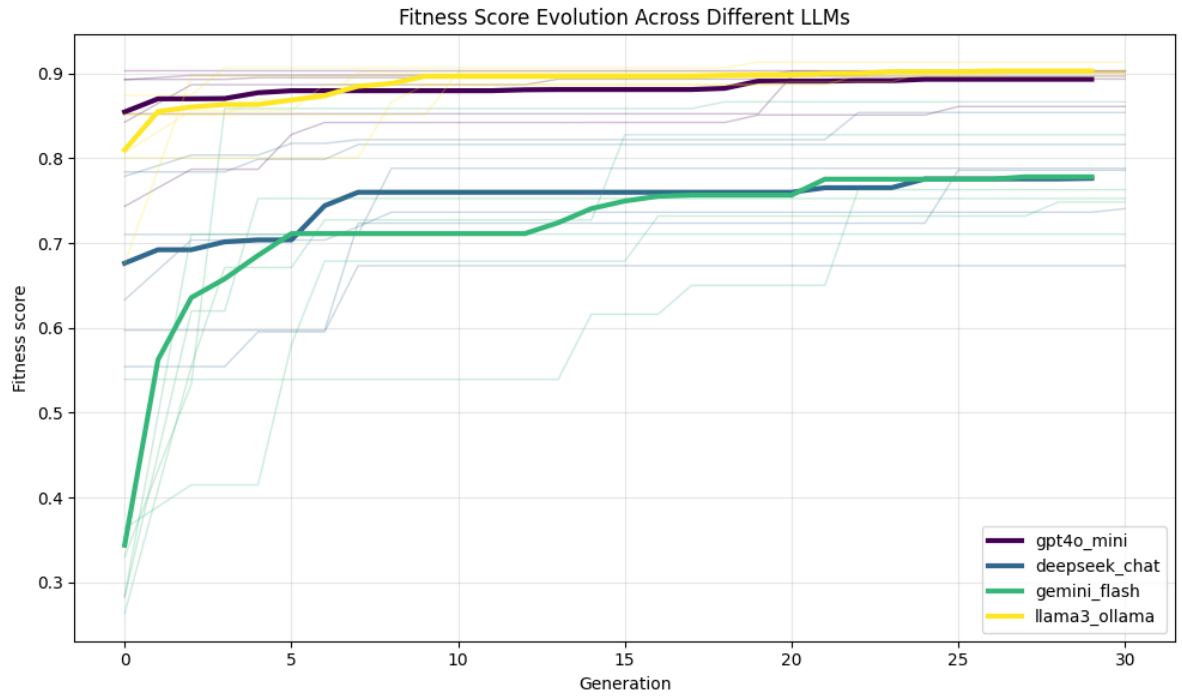

Figure 5.1: Figure illustrating the evolution of fitness scores across generations for four different large language models. Light-colored curves represent individual optimization runs, while bold curves indicate the average fitness progression per model.

Figure 5.1 shows the evolution of the fitness score across generations for all runs of the four evaluated language models. Individual runs are displayed using lighter-colored curves, while the bold curves represent the average fitness progression for each model.

Among the evaluated models, *gemini-flash* exhibits the steepest initial improvement in fitness, indicating that the optimization process is able to quickly exploit its generative capabilities during the early generations. However, despite this rapid initial progress, the model consistently converges to a plateau around a fitness score of approximately 0.8, suggesting premature convergence to a local optimum.

A similar behavior is observed for *deepseek-chat*. Although this model starts from a slightly higher initial fitness than *gemini-flash*, it ultimately converges to a comparable fitness plateau, indicating limited ability to escape local optima within the evolutionary loop.

In contrast, both *gpt-4o-mini* and *llama3* demonstrate superior optimization performance. These models start from relatively high initial fitness values and show steady, incremental improvements across generations, ultimately converging to fitness scores close to 0.9. This behavior suggests a stronger capacity for producing consistent refinements and benefiting from iterative feedback.

While *llama3* achieves a slightly higher final average fitness score than *gpt-4o-mini*, it requires significantly longer execution time due to its local deployment and higher computational cost per iteration. As a result, *gpt-4o-mini* was selected as the preferred model for subsequent experiments, as it offers a more favorable trade-off between solution quality and computational efficiency.

5.2 Experiment 2: Effect of mutation rate

Once the language model *gpt-4o-mini* is selected, the next step is to study the effect of different mutation ratios in order to analyze the exploration–exploitation trade-off of the optimization process.

As introduced previously, four mutation ratio values are evaluated. Figure 5.2 shows the fitness score evolution across generations for all runs of each mutation ratio. Individual runs are displayed in lighter colors, while the bold curves represent the average fitness progression.

For lower mutation ratios, in particular 0.0 and 0.3, the optimization process is dominated by refinement operations rather than large exploratory changes. In this regime, the model primarily performs incremental modifications to the current incumbent solution, resulting in small but consistent improvements. This behavior is clearly reflected in Figure 5.2, where the corresponding fitness curves exhibit a more pronounced and continuous increase during the early generations. However, once a fitness score of approximately 0.87–0.88 is reached, the curves plateau and show signs of premature convergence, indicating that the limited use of exploratory mutations reduces the ability of the system to escape local optima.

At the opposite extreme, a mutation ratio of 1.0, corresponding to always applying the redesign operator, results in slower and less stable improvement. In this setting, the fitness curve quickly reaches a plateau after only a few generations, suggesting that excessive exploration disrupts incremental progress. The lack of refinement prevents the system from consistently improving promising solutions, leading to stagnation in solution quality.

In contrast, an intermediate mutation ratio of 0.7 achieves a more favorable balance between exploration and exploitation. The corresponding fitness curve shows sustained improvement across generations and converges to the highest final average fitness score, close to 0.9. This indicates that combining redesign and refinement allows the system to both explore new solution structures and exploit high-quality candidates effectively.

Overall, these results confirm the importance of balancing mutation strategies in the proposed evolutionary framework. An intermediate mutation ratio enables continued progress throughout the optimization process, avoiding both premature convergence and excessive randomness, and is therefore adopted in subsequent experiments.

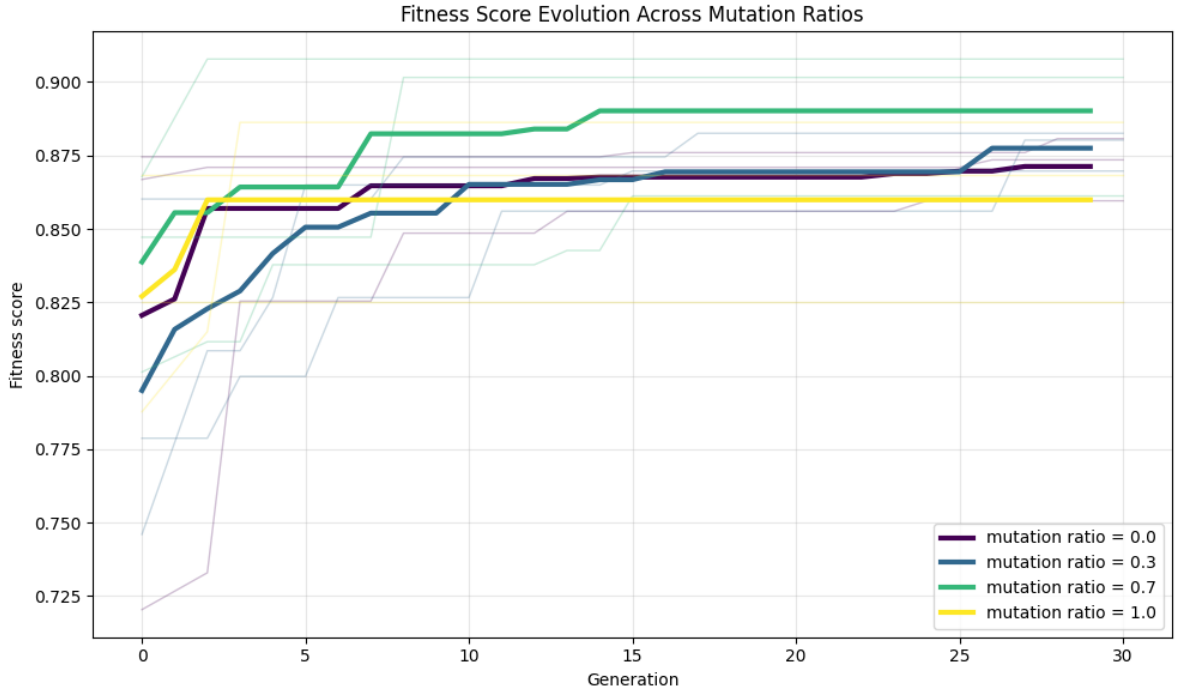

Figure 5.2: Figure illustrating the fitness score evolution across generations for different mutation ratios using the *gpt-4o-mini* model. Light-colored curves represent individual runs, while bold curves show the average fitness progression. The figure illustrates how different balances between redesign and refinement affect convergence behavior and final solution quality.

5.3 Experiment 3: Validity of LLM's predictions

To verify the validity of the heart rate predictions produced by the LLM, participants were asked to record their heart rate after each exercise and rest interval during the execution of two of the generated HIIT programs [1]. The recorded heart rate measurements are then compared against the predicted heart rate values. The resulting comparison figures for the corresponding HIIT sessions are illustrated in Figure 5.3 and Figure 5.4.

[1]The link to the videos of both HIIT programs is available here: `https://drive.google.com/drive/folders/1YKWUcZv6VBXUz4DqAlek-abX_Wnfkyjb?usp=sharing`

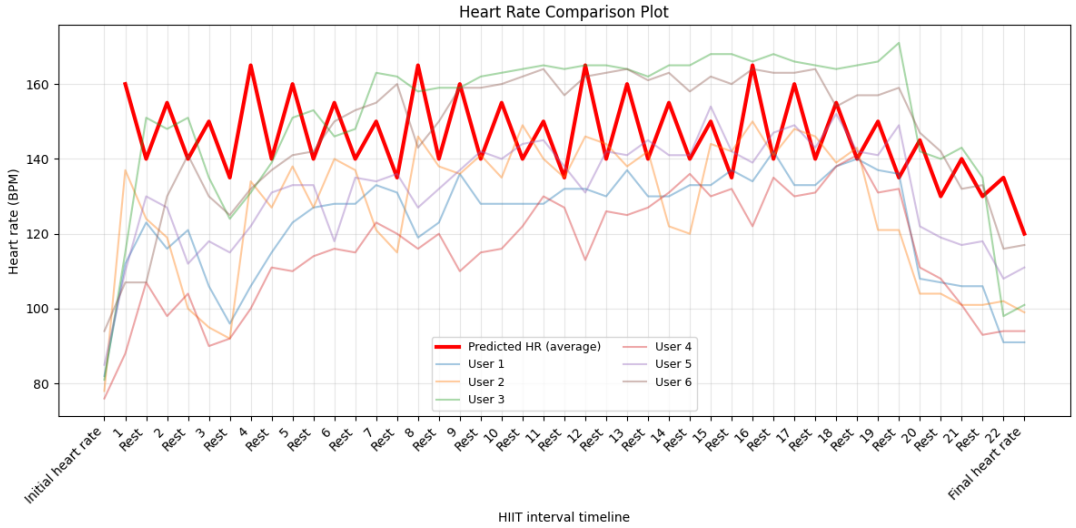

Figure 5.3: Comparison plot between LLM-predicted heart rate (bold line) and recorded user heart rates (lighter lines) across HIIT 1 session. The predicted average heart rate captures the general interval-based pattern but does not fully reflect the stabilization observed in the recorded measurements over time.

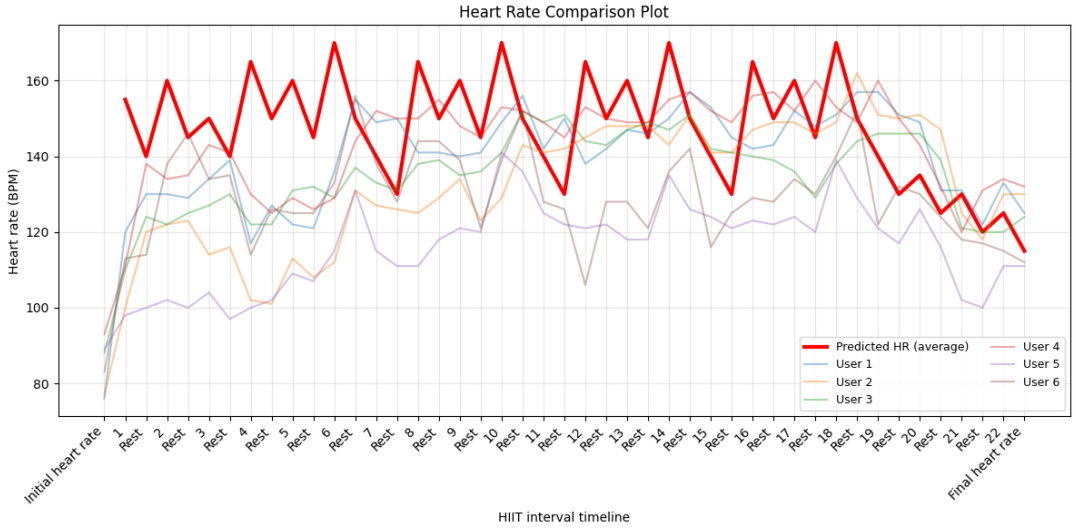

Figure 5.4: Comparison plot between LLM-predicted (bold line) and recorded heart rates (lighter lines) for HIIT 2 session. While the predicted curve follows the overall structure of work and rest intervals, it generally overestimates heart rate values relative to user measurements.

Overall, the results indicate that the LLM-generated predictions fall within a realistic range of heart rate values and exhibit reasonable variability over the course of a HIIT session. However, a closer analysis reveals several systematic discrepancies between predicted and actual recorded heart rates. The predictions do not consistently align with the true temporal dynamics. In particular, while the predicted curves exhibit regular oscillations that mirror the alternating structure of exercise and rest intervals, the phase of these oscillations often does not match the recorded measurements. In addition, in both figures, the predicted heart rate follows the same

oscillation pattern across the entire session, whereas the recorded heart rate tends to stabilize after a certain number of intervals. In reality, due to physiological adaptation and fatigue accumulation, the heart rate tends to gradually reach a steady elevated range. However, this stabilization effect is not sufficiently captured by the LLM predictions, which appear to treat each interval as largely independent of previous exertion.

Moreover, the predicted heart rates in the second HIIT program (Figure 5.4) are consistently higher than the actual values. These results suggest that the LLM may overestimate the heart rate. These discrepancies highlight a limitation of the current LLM-based heart rate estimation. While it produces plausible qualitative trends, it lacks a reliable model of effort accumulation, recovery dynamics, and individual variability. As a result, the predicted heart rate does not fully reflect the persistence of the exercise effect over time.

It is important to note that these measurements were obtained under non-controlled conditions, using consumer-grade heart rate tracking tools and self-reported recordings by participants. Consequently, the recorded data may contain noise and measurement inaccuracies. Nevertheless, the observed deviations remain consistent across sessions, indicating a genuine modeling limitation rather than purely experimental error.

Despite these limitations, the results demonstrate that the LLM-based prediction provides a reasonable first-order approximation of heart rate dynamics. A clear direction for future work would be improving the accuracy of these predictions by incorporating fatigue modeling, user-specific calibration, or physiological constraints. A more accurate heart rate estimation would directly improve the realism of the generated HIIT programs and enhance the fitness function, leading to better alignment between optimized routines and real-world physiological responses.

5.4 Experiment 4: User feedback

To assess how accurately the proposed fitness function 3.1 captures the perceived quality of a HIIT program, we conducted a user-based evaluation involving the workouts from the previous experiment that received markedly different fitness scores from the system. The goal of this evaluation is to compare the system's internal assessment with human judgments.

In order to have a fair comparison, participants who performed both HIIT programs are asked to evaluate in a scale from $1 - 10$ each workout according to the same components used in the fitness formulation, namely power demand, work–rest balance, perceived intensity, and exercise variety. The heart-rate component is treated separately and is not subjectively rated; instead, it is computed from the physiological data collected during the previous experiment using user-recorded heart rates. This ensures that the evaluation remains consistent with the structure of the fitness function while avoiding unreliable subjective estimates of heart rate.

Tables 5.1a and 5.1b report the average user scores for each component, together with the resulting fitness values obtained by applying the proposed formulation. The individual ratings provided by each participant are reported in Appendix A for completeness.

The fitness scores generated by the system for the two HIIT programs are 0.890 for HIIT 1 and 0.387 for HIIT 2. When recomputed using user-averaged component scores and user-measured heart rate, the relative ordering remains consistent, with HIIT 1 being evaluated with a higher quality than HIIT 2, both by the system and by human users. Although the gap between the

Component	Score
Heart rate	0.655
Power	0.707
Work–rest balance	0.999
Intensity	0.998
Variety	0.633
Final fitness s	0.786
Final fitness (0–10)	7.86
System score	0.890

(a) HIIT Program 1

Component	Score
Heart rate	0.621
Power	0.946
Work–rest balance	0.820
Intensity	0.770
Variety	0.666
Final fitness s	0.769
Final fitness (0–10)	7.69
System score	0.387

(b) HIIT Program 2

Table 5.1: Comparison of fitness scores for the respective HIIT programs based on user evaluations and measured heart rate. The final score aligns with the system's fitness assessment, with HIIT 1 receiving the highest score.

user-based fitness scores is smaller than the one produced by the system, this result suggests that the fitness function captures meaningful qualitative differences between workouts rather than producing arbitrary rankings.

In the designed scoring mechanism, higher component scores indicate closer alignment between user evaluations and a predefined target range, which represents the desired or optimal value for each feature. Conversely, lower scores arise when user evaluations deviate substantially from this range, either by being too low or too high. This behavior results from the use of Gaussian-shaped scoring functions, which are designed to reward values near the target while progressively penalizing deviations in both directions.

This effect can be clearly observed in the case of the power demand component. Although users rated the power requirement of HIIT 1 relatively high (see Appendix A), these ratings exceeded the preferred range implicitly encoded in the fitness function, which is centered around a normalized value of 0.6. As a consequence, the corresponding Gaussian score is lower than that of HIIT 2, whose power demand evaluations fall closer to the target range. This example highlights a discrepancy between predefined parameter targets and subjective user perception, emphasizing the importance of carefully calibrating scoring functions to reflect realistic human expectations.

Overall, these results indicate that the proposed fitness function constitutes a promising first approximation of HIIT program quality. While it cannot be claimed that the formulation is universally optimal or applicable to all workout styles and user populations, it demonstrates the ability to align system-based evaluations with human judgments in a consistent and interpretable manner.

5.5 Experiment 5: Trainer ABtest

In this last experiment, a qualitative assessment is conducted with four professional trainers to evaluate the *ABtest* mode of the system from an expert perspective. The trainers are asked to interact with the system to generate a HIIT program appropriate for women between the

Evaluation criterion	Average score
Professionalism of generated HIIT routines	5.75
Accuracy of response to trainer feedback	4.50
Diversity of generated routines	5.00
Overall quality of the final HIIT program	5.50

Table 5.2: Average evaluation scores provided by professional trainers for the final HIIT program across multiple assessment criteria, using a 1–10 rating scale.

age of $20 - 30$ without requiring expertise in exercising. Then, they need to evaluate the final output from four perspectives: the perceived professionalism of the generated workouts, the system's responsiveness to expert feedback, the diversity of generated routines, and the overall quality of the final HIIT program.

The results are presented in Table 5.2 as the average score for each of the components from all four trainers in a scale of $1 - 10$. As reported in the table, the highest score corresponds to the professionalism of the generated routines, suggesting the system is capable of producing workouts that exhibit a reasonable level of structure and coherence, although there remains substantial room for improvement before reaching professional-grade standards. Similarly, the overall quality of the final program received a moderate score, indicating that while the output is acceptable, it does not yet fully meet expert expectations. The lowest score was assigned to the accuracy of the system's response to trainer feedback. This result indicates that, although feedback is incorporated to some extent, the system does not always reflect expert guidance in a sufficiently precise or predictable manner. This limitation highlights a key challenge for human-in-the-loop optimization with LLMs, particularly when expert feedback is nuanced or domain-specific.

In addition to the quantitative ratings, trainers are also asked whether they would prefer to rely on the system for generating HIIT workouts or continue designing programs entirely manually. Interestingly, only one trainer expressed a preference for fully human-driven design, whereas the remaining participants favored a hybrid approach combining system-generated suggestions with human expertise and final decision-making. This outcome suggests that the proposed framework may be most effective not as a replacement for professional trainers but as a collaborative tool that supports human creativity and expertise.

Taken together, these findings indicate that a hybrid workflow may constitute a more practical and acceptable solution in real-world training contexts. This conclusion suggests that our *Hybrid* model emerges as a promising candidate for use by professional coaches.

Chapter 6

Conclusion

6.1 Reflection on research questions

After conducting all the experiments and collecting both quantitative and qualitative feedback, we can now reflect on the research questions formulated at the beginning of this thesis.

RQ1. How to design a numerical formulation suitable for evolutionary optimization that can effectively represent the quality of a HIIT program?

The primary objective of this work was to design a system capable of generating HIIT programs within an automated optimization loop. To achieve this, we proposed an approach that integrates LLMs into an evolutionary framework, enabling the structured generation and iterative refinement of HIIT workout routines. A central challenge in this process is therefore the design of a fitness function able to represent the quality of a HIIT program in an objective and professionally meaningful way.

In practice, HIIT programs are inherently context-dependent and often tailored to specific populations based on physiological characteristics, fitness levels, and training goals. For this reason, the proposed fitness function is designed for a clearly defined target group: women aged between 20 and 30 years. The function aims to capture multiple dimensions that professional coaches typically consider when designing HIIT routines, including physiological responses, biomechanical constraints, structural coherence, intensity distribution, and exercise variety within controlled ranges.

The experimental results indicate that the proposed fitness function constitutes a promising first attempt at numerically representing HIIT quality. While the formulation is not intended to be universal or definitive, it demonstrates that complex, qualitative coaching criteria can be translated into a structured numerical representation. This suggests that further exploration of numerical fitness formulations for workout quality is a valuable research direction. The core idea of the research would be focused on encoding expert reasoning into quantitative metrics.

RQ2. To what extent can an LLM be effectively integrated into an evolutionary framework for structured workout generation?

The integration of an LLM into the evolutionary framework is definitely a powerful design choice. Beyond the specific application of workout generation, this work highlights the broader

potential of combining LLMs with evolutionary algorithms for structured content creation tasks. The LLM provides high-level reasoning, creativity, and linguistic flexibility, while the evolutionary process enforces structure, selection pressure, and iterative optimization.

Throughout the project, we clearly identified and verified with professional coaches that the LLM often generated innovative and unconventional combinations of exercises. Several routines were described by experts as creative and novel, yet still realistic and potentially effective. This suggests that the LLM does not merely replicate existing patterns but can meaningfully expand the design space while remaining within practical constraints when guided by an appropriate fitness function. These findings support the effectiveness of the integration of LLM into evolutionary algorithms for generating structured, high-quality workout programs.

RQ3. To what extent is an evolutionary LLM-based HIIT generation system acceptable and useful to human users in practice?

Regarding practical usefulness and acceptance, the results suggest that while some coaches still prefer a fully manual HIIT design, the majority of professionals expressed a willingness to incorporate the system into their workflow. Rather than replacing human expertise, the system was primarily valued as a collaborative and inspirational tool. Coaches indicated that it could support idea inspiration, routine variation, and long-term planning, while still allowing human judgment and adaptation.

For non-professional users, the system also demonstrated practical relevance. Although it cannot replace the nuanced expertise of a trained coach, the generated programs were consistently evaluated as realistic, executable, and beneficial. This indicates that the system has the potential to lower the barrier to structured HIIT training while maintaining a reasonable level of safety and effectiveness.

Overall, these findings suggest that an evolutionary LLM-based HIIT generation system is both acceptable and useful in real-world contexts, particularly when positioned as a hybrid tool that combines automated generation with human oversight. Such collaboration appears to be a promising and practical direction for future fitness-oriented AI systems.

6.2 Limitations and challenges

Despite the promising results obtained in this work, several limitations and challenges were identified throughout the project, which should be addressed in future research.

One of the main limitations concerns the accuracy of the physiological predictions produced by the LLM. In Experiment 3, users' actual heart rate measurements were collected and compared against the system's predictions. While the predicted trends were generally consistent with the observed data, the results were not fully accurate, leaving a non-negligible margin of error. In addition, due to the lack of appropriate measurement tools, the predicted power output could not be empirically validated. Although these inaccuracies did not critically compromise the qualitative assessment of the generated HIIT programs, they may nonetheless affect the reliability of the fitness function and, consequently, the selection pressure within the evolutionary optimization process. Improving the precision of physiological predictions represents an important direction for future work.

A second limitation relates to the expressiveness and diversity of the HIIT structures generated

by the system. Feedback from professional coaches indicated that the representation of HIIT formats was somewhat restrictive. In particular, the system predominantly focused on short, time-based intervals, whereas HIIT in practice can be implemented through a wider range of structures. These include, for example, completing a fixed number of repetitions as quickly as possible within a given time window, followed by structured recovery periods, or longer high-intensity efforts exceeding one minute, depending on training goals. As a result, the variation in HIIT formats was perceived as limited and somewhat one-sided. While these constraints were necessary to ensure evaluability and comparability during optimization, they also reduced the space of possible workout designs. Expanding the representation of HIIT structures and incorporating additional format types would increase both the realism and applicability of the generated programs, particularly for professional training contexts.

6.3 Future work

The limitations identified in this thesis open several promising directions for future research and development. Addressing these aspects would not only strengthen the technical robustness of the system but also expand its applicability in real-world fitness and coaching contexts.

A first important direction concerns the improvement of physiological prediction accuracy. Future work could integrate hybrid approaches combining LLM-based reasoning with data-driven physiological models could be explored to better estimate heart rate dynamics and power output across different workout structures and user profiles. This would allow the system to rely on more a more reliable data base, reducing systematic bias and improving reliability. Such improvements would directly enhance the effectiveness of the fitness function and the evolutionary selection process.

A second direction involves extending the expressiveness and diversity of HIIT structures supported by the system. Future versions could incorporate a broader range of HIIT formats, including repetition-based protocols, task-completion challenges, and longer high-intensity efforts combined with structured recovery phases. This would require both an extension of the workout representation and a reformulation of the fitness function to fairly evaluate heterogeneous training formats. Expanding the design space in this way would improve realism and make the system more aligned with professional coaching practices.

Another promising avenue is personalization. While this work focused on a specific target group, future research could generalize the framework to support multiple user profiles with varying age ranges, fitness levels, and training goals. Adaptive fitness functions or multi-objective optimization strategies could be employed to provide realistic workout routines adapted to each user's fitness objectives. This would allow the system to evolve workouts that are not only high-quality but also individually tailored.

Taken together, these results suggest that the proposed LLM-based evolutionary system is a promising approach for HIIT program generation, opening several directions for future work. With improved physiological modeling, greater structural diversity, enhanced personalization, and deeper human collaboration, such systems have the potential to become valuable tools in both professional training environments and personal fitness applications.

Bibliography

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[2] Milad Asgari Mehrabadi, Elahe Khatibi, Tamara Jimah, Sina Labbaf, Holly Borg, Pamela Pimentel, Nikil Dutt, Yuqing Guo, and Amir M. Rahmani. Perfect: Personalized exercise recommendation framework and architecture. *medRxiv preprint*, 2023.

[3] Muhammed Mustafa Atakan, Yanchun Li, Şükran Nazan Koşar, Hüseyin Hüsrev Turnagöl, and Xu Yan. Evidence-based effects of high-intensity interval training on exercise capacity and health: A review with historical perspective. *International Journal of Environmental Research and Public Health*, 18(13):7201, 2021.

[4] Adam P Balcerzak, Marek Zinecker, and Jiří Mičánek. The impact of artificial intelligence on task performance and decision-making: Empirical evidence on generation z. *Human Technology*, 21(3):620–639, 2025.

[5] Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. Deepcoder: Learning to write programs. In *International Conference on Learning Representations (ICLR)*, 2017.

[6] Thomas Bartz-Beielstein, Jürgen Branke, Jörn Mehnen, and Olaf Mersmann. Overview: Evolutionary algorithms. In Thomas Bartz-Beielstein, Wolfgang Konen, Horst Stenzel, and Boris Naujoks, editors, *Schriftenreihe CIplus*, volume 2. CIplus, 2015. Band 2/2015.

[7] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big?. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

[8] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[9] Rishi Bommasani. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[10] Tudor O Bompa and Carlo Buzzichelli. *Periodization-: theory and methodology of training*. Human kinetics, 2019.

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Lan-

guage models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[12] Dikshit Chauhan, Bapi Dutta, Indu Bala, Niki van Stein, Thomas Bäck, and Anupam Yadav. Evolutionary computation and large language models: A survey of methods, synergies, and applications. *arXiv preprint arXiv:2505.15741*, 2025.

[13] CrossFit, LLC. Crossfit terms explained, 2026.

[14] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. RLPROMPT: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[15] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Springer, 2015.

[16] Evo Fitness. Tabata, emom, amrap & hiit: What is the difference?, 2026.

[17] Anthony Favier, Ngoc La, Pulkit Verma, and Julie A. Shah. An llm-powered collaborative task planning framework. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), System Demonstrations Track*, 2025.

[18] Yu Feng, Shiqing Wang, Yichi Zhang, and Jie Tang. Evolution of heuristics: Towards efficient automatic algorithm design using large language models. *arXiv preprint arXiv:2401.02051*, 2024.

[19] Tim J Gabbett. The training—injury prevention paradox: should athletes be training smarter and harder? *British journal of sports medicine*, 50(5):273–280, 2016.

[20] Sumit Gulwani. Automating string processing in spreadsheets using input-output examples. *ACM Sigplan Notices*, 46(1):317–330, 2011.

[21] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2307.08541*, 2023.

[22] Yiduo Guo, Yaobo Liang, Chenfei Wu, Wenshan Wu, Dongyan Zhao, and Nan Duan. Learning to plan by updating natural language. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

[23] Nikolaus Hansen and Raymond Ros. Black-box optimization benchmarking of NEWUOA compared to BIPOP-CMA-ES: On the BBOB noiseless testbed. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 1519–1526. ACM, 2010.

[24] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[25] Qin Jianjun, Haytham F Isleem, Walaa JK Almoghayer, and Mohammad Khishe. Predictive athlete performance modeling with machine learning and biometric data integration. *Scientific Reports*, 15(1):16365, 2025.

[26] Hyston Kayange, Jonghyeok Mun, Yohan Park, Jongsun Choi, and Jaeyoung Choi. A hybrid approach to modeling heart rate response for personalized fitness recommendations using wearable data. *Electronics*, 13(19):3888, 2024.

[27] Justin Khasentino, Anastasiya Belyaeva, Xin Liu, Zhun Yang, Nicholas A. Furlotte, Chace Lee, Erik Schenck, Yojan Patel, Jian Cui, Logan Douglas Schneider, Robby Bryant, Ryan G. Gomes, Allen Jiang, Roy Lee, Yun Liu, Javier Perez, Jameson K. Rogers, Cathy Speed, Shyam Tailor, Megan Walker, Jeffrey Yu, Tim Althoff, Conor Heneghan, John Hernandez, and Cory Y. McLean. A personal health large language model for sleep and fitness coaching. *Nature Medicine*, 31:3394–3403, 2025.

[28] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[29] Marco Laumanns, Günter Rudolph, and Hans-Paul Schwefel. Mutation control and convergence in evolutionary multi-objective optimization. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO)*, volume 1993 of *Lecture Notes in Computer Science*, pages 176–190. Springer, 2001.

[30] Paul B Laursen and David G Jenkins. The scientific basis for high-intensity interval training. *Sports Med*, 32(1):1, 2001.

[31] Wenjun Li, Changyu Chen, and Pradeep Varakantham. Unlocking the planning capabilities of large language models with maximum diversity fine-tuning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3318–3340, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics.

[32] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, John Agapiou, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[33] Yujia Li et al. Eureka: Human-level skill discovery via LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[34] Manuel López-Ibáñez, Diederick Vermetten, Johann Dréo, and Carola Doerr. Using the empirical attainment function for analyzing single-objective black-box optimization algorithms. *arXiv preprint arXiv:2404.02031*, 2024.

[35] Jianmo Ni, Larry Muhlstein, and Julian McAuley. Modeling heart rate and activity data for personalized fitness recommendation. In *Proceedings of the World Wide Web Conference (WWW)*, pages 1145–1155. ACM, 2019.

[36] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[37] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[38] Bernardino Romera-Paredes et al. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024.

[39] Donghoon Shin, Gary Hsieh, and Young-Ho Kim. Planfitting: Personalized exercise planning with large language model-driven conversational agent. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI)*, 2024.

[40] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[41] Izumi Tabata, Kouichi Nishimura, Motoki Kouzaki, Yosuke Hirai, Fumio Ogita, Motohiko Miyachi, and Kazuhisa Yamamoto. Effects of moderate-intensity endurance and high-intensity intermittent training on anaerobic capacity and VO_2max. *Medicine & Science in Sports & Exercise*, 28(10):1327–1330, 1996.

[42] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *arXiv preprint arXiv:2206.10498*, 2023.

[43] Sander van Rijn, Hao Wang, Matthijs van Leeuwen, and Thomas Bäck. Evolving the structure of evolution strategies. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.

[44] Niki van Stein and Thomas Bäck. Llamea: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 2024.

[45] Diederick Vermetten, Fabio Caraffini, Anna V Kononova, and Thomas Bäck. Modular differential evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 864–872, 2023.

[46] Diederick Vermetten, Manuel López-Ibáñez, Olaf Mersmann, Richard Allmendinger, and Anna V Kononova. Analysis of modular cma-es on strict box-constrained problems in the sbox-cost benchmarking suite. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 2346–2353, 2023.

[47] Wikipedia contributors. High-intensity interval training, 2026.

[48] Wikipedia contributors. Normal distribution, 2026.

[49] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[50] Wenshuo Zhai, Jinzhi Liao, Ziyang Chen, Bolun Su, and Xiang Zhao. A survey of task planning with large language models. *Intelligent Computing*, 4:0124, 2025.

[51] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations (ICLR)*, 2023.

Appendix A

Additional experimental results

User	Power	Work–rest	Intensity	Variety	Overall quality
U1	8	7	7	8	9
U2	7	9	7	5	8
U3	8	8	6	7	7
U4	7	5	8	5	7
U5	7	8	8	6	7
U6	8	4	8	4	7
U7	8	5	8	7	8
U8	8	6	8	7	7
U9	8	6	8	8	8
Average	7.667	6.444	7.556	6.333	7.556

Table A.1: User evaluation scores for HIIT program 1 across the defined subjective criteria in a 1–10 scale.

User	Power	Work–rest	Intensity	Variety	Overall quality
U1	8	9	8	5	8
U2	8	9	8	9	9
U3	6	8	7	7	7
U4	4	6	5	6	6
U5	8	8	8	7	8
U6	8	5	8	5	7
U7	7	7	7	6	8
U8	6	7	6	7	8
U9	5	8	4	8	7
Average	6.667	7.444	6.778	6.667	7.556

Table A.2: User evaluation scores for HIIT Program 2 across the defined subjective criteria in a 1–10 scale.