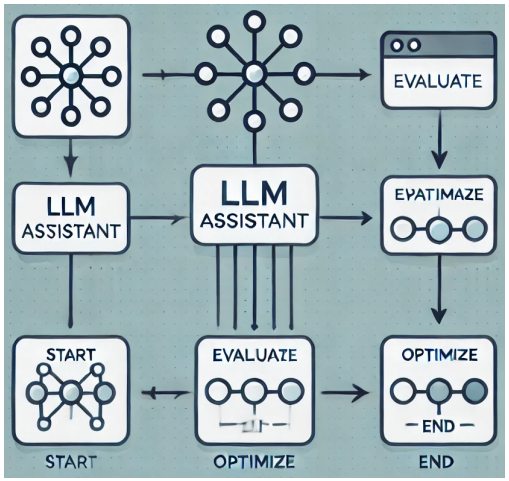
Optimizing the Optimizer: An Example Showing the Power of LLM Code Generation (FedCSIS 2025, Invited Contribution)

© Camilo Chacón Sartori & **Christian Blum**

Artificial Intelligence Research Institute (IIIA-CSIC)



Personal introduction: IIIA-CSIC

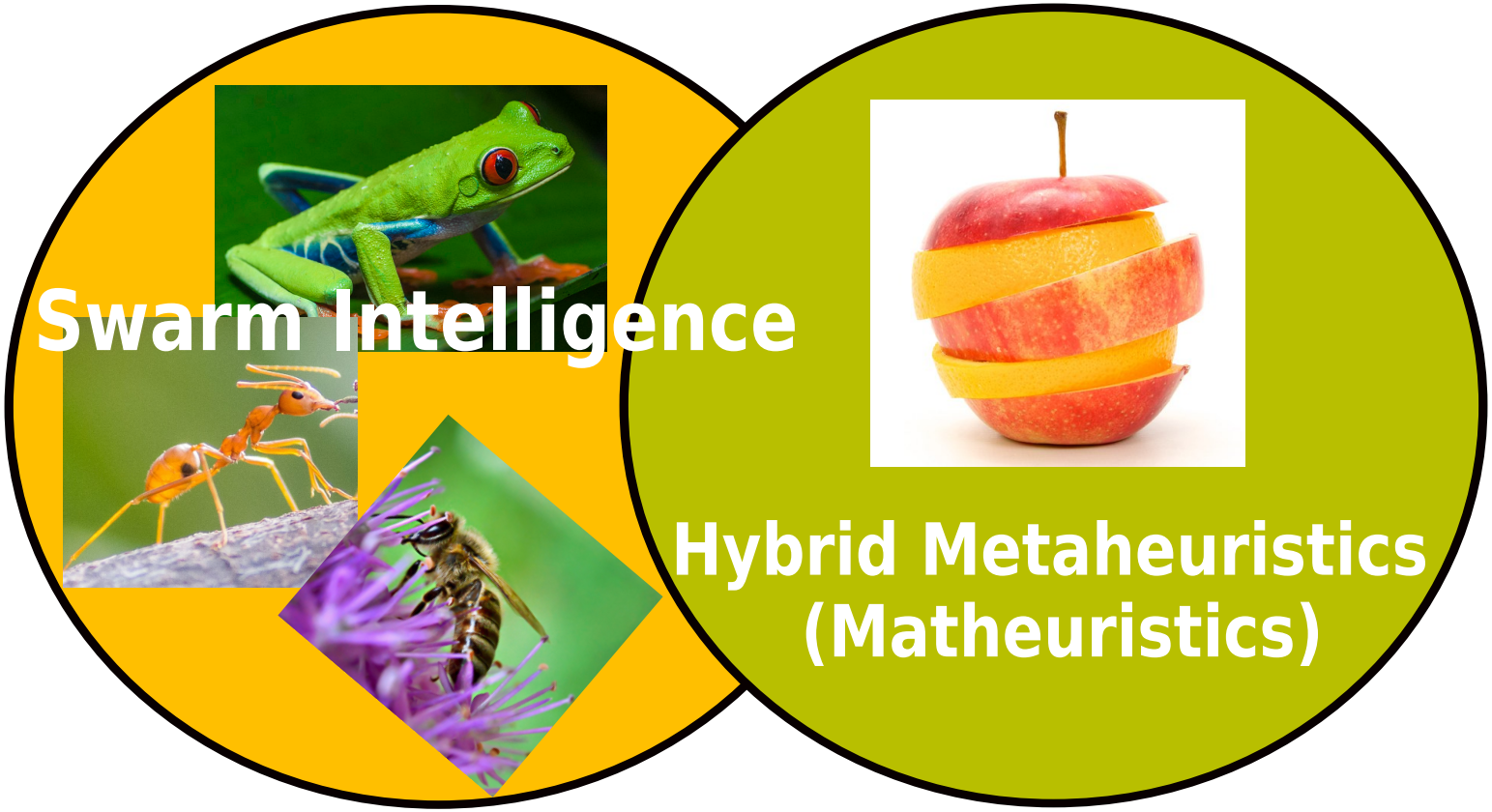
CSIC: Spanish National Research Council

- Largest public institution dedicated to research in Spain (created in 1939)
- Third-largest in Europe
- 6% of all research staff in Spain work for the CSIC
- 20% of the scientific production in Spain is from the CSIC

Artificial Intelligence Research Institute (IIIA)

- 28 tenured scientists (of three different ranks)
- Around 50 additional staff member (post-docs, PhD students, technicians, administration)
- Research lines: machine learning, optimization, logic and reasoning, multi-agent systems

Our research topics: algorithm-oriented



My main objective

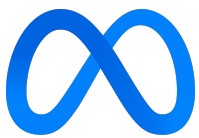
- **Basic research:** general optimization algorithms
- **Applied research:** optimization problems on sustainable development

What are Large Language Models?

- LLMs are a type of artificial intelligence model designed to understand and generate human language.
- Built using deep learning techniques, particularly Transformer architectures.
- Trained on massive corpora of text data to learn patterns, grammar, facts, and reasoning abilities.
- **Examples:**


OpenAI's GPT


Google's Gemini


Meta's LLaMA


Anthropic's Claude


DeepSeek

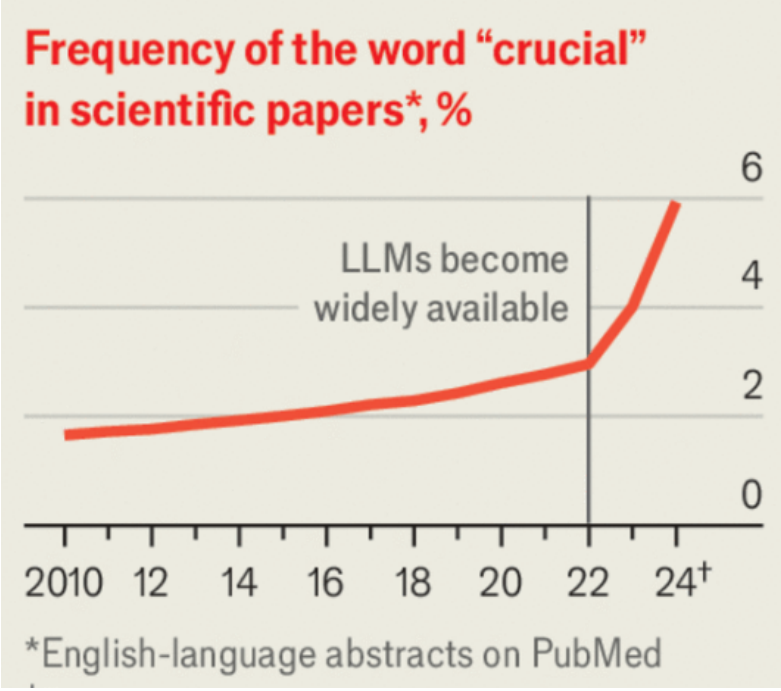
Key Features of LLMs

- **They are huge**: billions to trillions of parameters.
- **Pretraining and fine-tuning**: trained on general data, but they can also be adapted to specific tasks.
- **Zero-shot, One-shot, Few-shot**: Capable of performing tasks with minimal examples.[1]
- **Multitask capability**: Can handle translation, summarization, question answering, coding, and more.

[1] **Example for zero-shot:** You ask a model, Is the following sentence sarcastic? and provide a sentence. If it answers correctly, despite never being trained on sarcasm detection examples, thats zero-shot.

Applications of LLMs

- Text generation and completion
- Chatbots and conversational AI
- Code generation and debugging
- Language translation
- Content summarization
- **But also lately: Mathematical reasoning**[a]

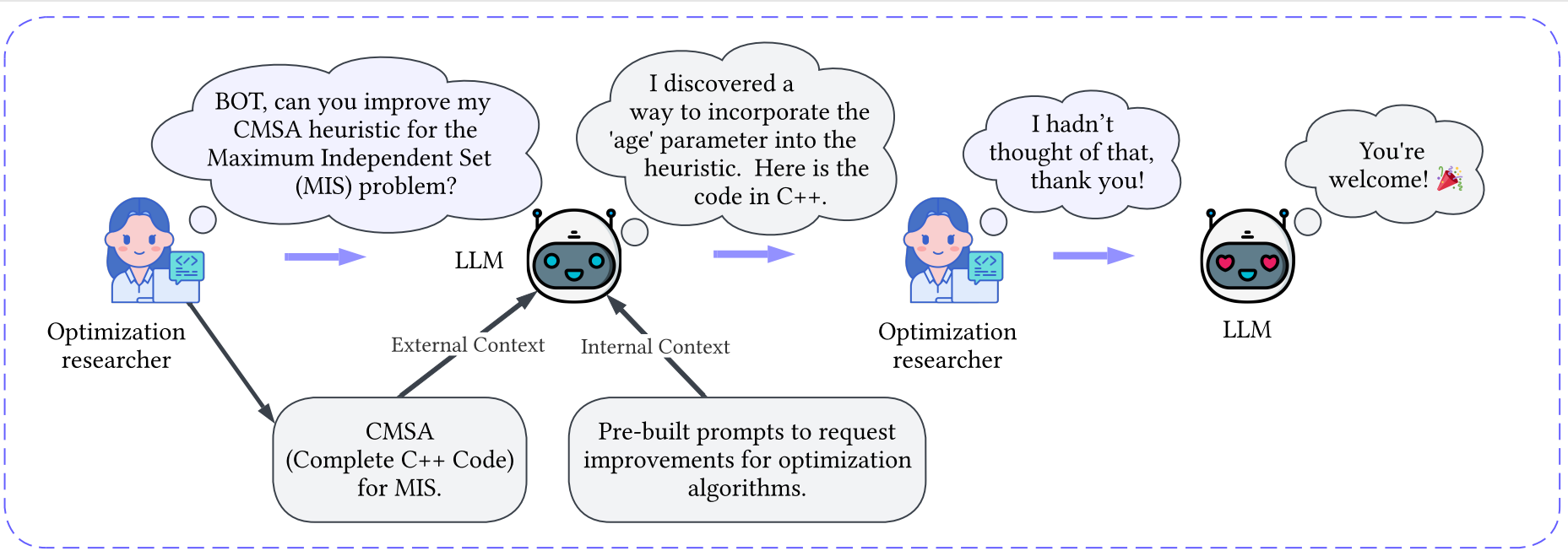


©The Economist

[a] Ahn, J., Verma, R., Lou, R., Liu, D., Zhang, R., & Yin, W. (2024). Large language models for mathematical reasoning: Progress and challenges. arXiv preprint arXiv:2402.00157

Idea: Improve Expert Optimization Code

Improving Expert-Designed Algorithms with LLMs

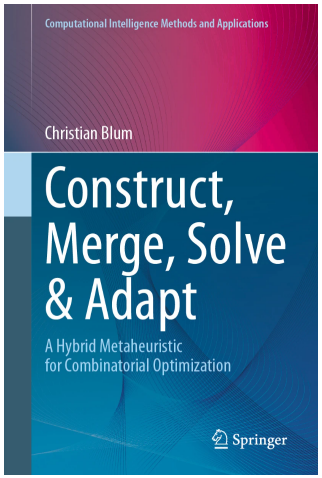


https://www.arxiv.org/abs/2502.08298

To be published in proceedings of FedCSIS 2025

Contruct, Merge, Solve & Adapt, CMSA

Initial CMSA team

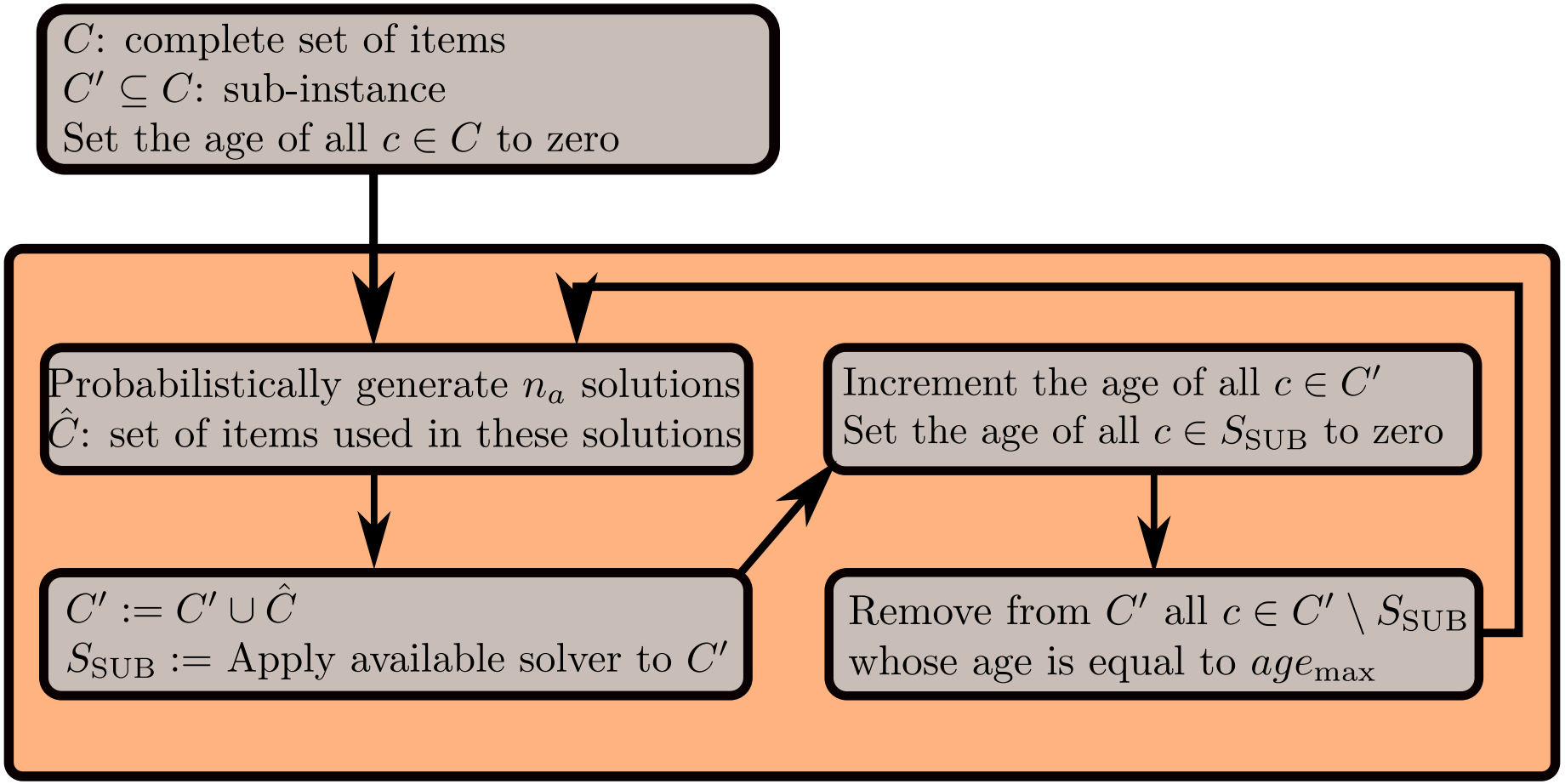


CMSA Book 2024

Winners of the **SEIO-FBBVA award 2021** for the ***best methodological contribution to Operations Research***.

- Spanish Society of Statistics and Operations Research
- Fundación BBVA BBVA Foundation (Banco Bilbao Vizcaya Argentaria)

CMSA: flow diagram

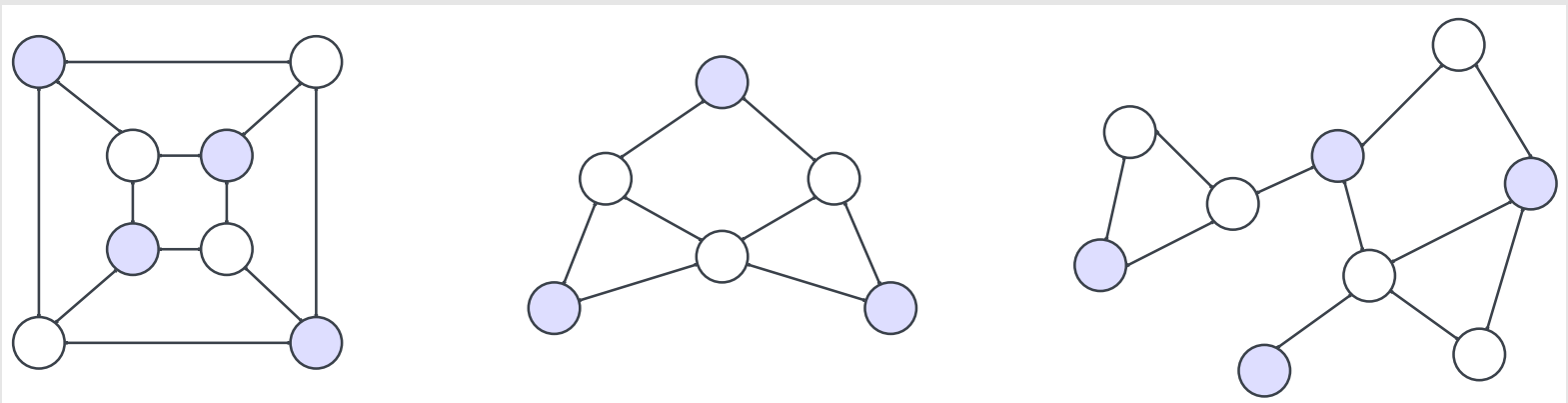


Optimization Problem: MIS

Definition: Maximum Independent Set (MIS) problem

- **Given:** An undirected graph $G = (V, E)$.
- **Search space:** Every set $S \subseteq V$ such that for every $v \in S$ it holds that $N(v) \cap S = \emptyset$
- **Objective function:** $F(S) := |S|$ (cardinality of set S)
- **Optimization objective:** maximization

Examples of solutions



CMSA: Algorithm Components

ILP model of the MIS Problem

$$\begin{aligned} \max \quad & \sum_{v \in V} x_v \\ \text{subject to} \quad & x_v + x_u \leq 1 \qquad \forall\, (x_v, x_u) \in E \\ & x_v \in \{0,1\} \qquad \forall\, v \in V \end{aligned}$$

What do we need for implementing a CMSA?

1. Define the **set of solution components**
2. A **greedy heuristic** used in a probabilistic way for generating solutions
3. An **approach for solving the sub-instance** of each iteration:
   - An exact technique
   - A metaheuristic approach

CMSA application to the MIS

Definition of solution components

For or each vertex v_i we introduce a solution component c_i

MIS Greedy Heuristic

1. Start with an empty solution S
2. Add exactly one feasible vertex to S at each construction step. Feasible vertices have no neighbors in S.
3. Choice of a feasible vertex:
   - **With probability** $r < 1$**:** Choose feasible vertex with minimum degree
   - **Otherwise:** Choose a feasible vertex by roulette-wheel-selection according to these weights:

$$w(v) := \frac{1}{|N(v)| + 1} \qquad \forall\, v \in V \text{ that are feasible}$$

Prompt conversation with GPT

First Prompt

What improvements can I make to my CMSA implementation for solving the Maximum Independent Set (MIS)?

Current implementation:
{{CMSA algorithm code in C++ of ca. 400 lines}}

GPT Answer

Your implementation of the Construct, Merge, Solver, and Adapt (CMSA) for solving the Maximum Independent Set (MIS) problem appears to be a solid base. However, there are several improvements that could be made to enhance performance, ...

2. Algorithmic Improvements
Degree-based heuristic: You're using an increasing degree order (likely vertices with lower degrees are selected first). This is a common heuristic, **but you might also want to experiment with other heuristics, such as selecting vertices based on a combination of degree and the age of vertices** (e.g., vertices with low degrees and a low age could be more promising)}. This would add a more dynamic element to your heuristic. ...

Second Prompt

```
Please consider the "age" parameter when designing the new heuristic of
      CMSA, which you find in the function named
      generate_solution(...). Provide me with the C++ code of your
      newly designed heuristic.
```

LLM-Suggested MIS Greedy Heuristic

1. Choice of a feasible vertex:
   - **With probability** $r < 1$**:** Choose feasible vertex with minimum degree
   - **Otherwise:** Choose a feasible vertex by roulette-wheel-selection according to these weights:

$$w(v) := \frac{1}{|N(v)|+1} + \cancel{\frac{1}{age[v]+1}} + \frac{1}{age[v]+2} \qquad \forall\, v \in V \text{ that are feasible}$$

Prompt conversation with GPT

Third Prompt

```
Are there ways to enhance the dynamic vertex selection mechanism to
allow for a more diverse and advanced search process?
```

Improved LLM-Suggested MIS Greedy Heuristic

Make use of entropy-adjusted vertex choice probabilities:

1. **Previous probabilities:** $P(v) := \frac{w(v)}{\sum_{v' \text{ feasible}} w(v')}$
2. **Entropy-adjusted:** $P_H(v) := \frac{P(v)+H}{\sum_{v' \text{ feasible}} P(v') + H}$, where

$$H = -\sum_{v'' \in V} P(v'')\log(P(v''))$$

Asking the LLM for Code Improvement

Obtained CMSA Variants

1. CMSA: Standard CMSA
2. LLM-CMSA-V1: CMSA using "age"-value-extended heuristic
3. LLM-CMSA-V2: Entropy-adjusted variant of LLM-CMSA-V1

Fourth Prompt

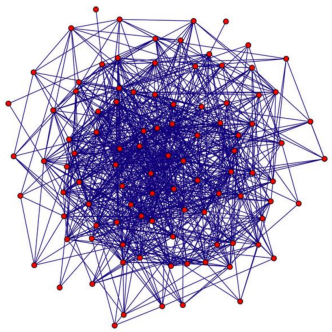
Given the following heuristic implemented in C++ in function generate_solution(...), without altering its core logic or functionality, please analyze the code to identify potential **improvements in the use of data structures, cache optimization, and other low-level optimizations. Focus on enhancing performance by suggesting more efficient data structures, reducing memory overhead, improving data locality, and leveraging modern C++ features where applicable.** Please provide an updated version of the code with comments explaining each optimization.

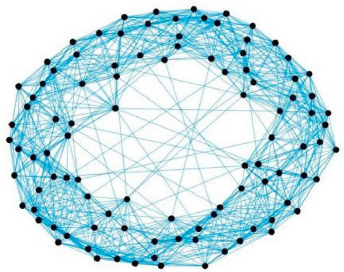
Experimental Setup

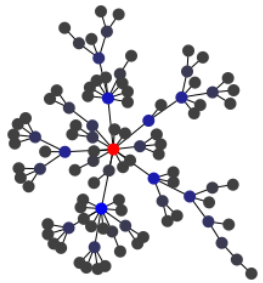
Problem Instances, Algorithms, and Tuning

- **Problem instances:** 480 **Erdös-Rényi** graphs, 480 **Watts-Strogatz** graphs, 480 **Barabási-Albert** graphs. The three sets contain graphs with 500-2000 vertices and different densities.
- **Algorithm codes:** 5 different algorithm codes
- **Algorithm tuning:** The 5 algorithm codes were tuned with the scientific tuning software `irace`[a]

[a] López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives, 3, 43-58.

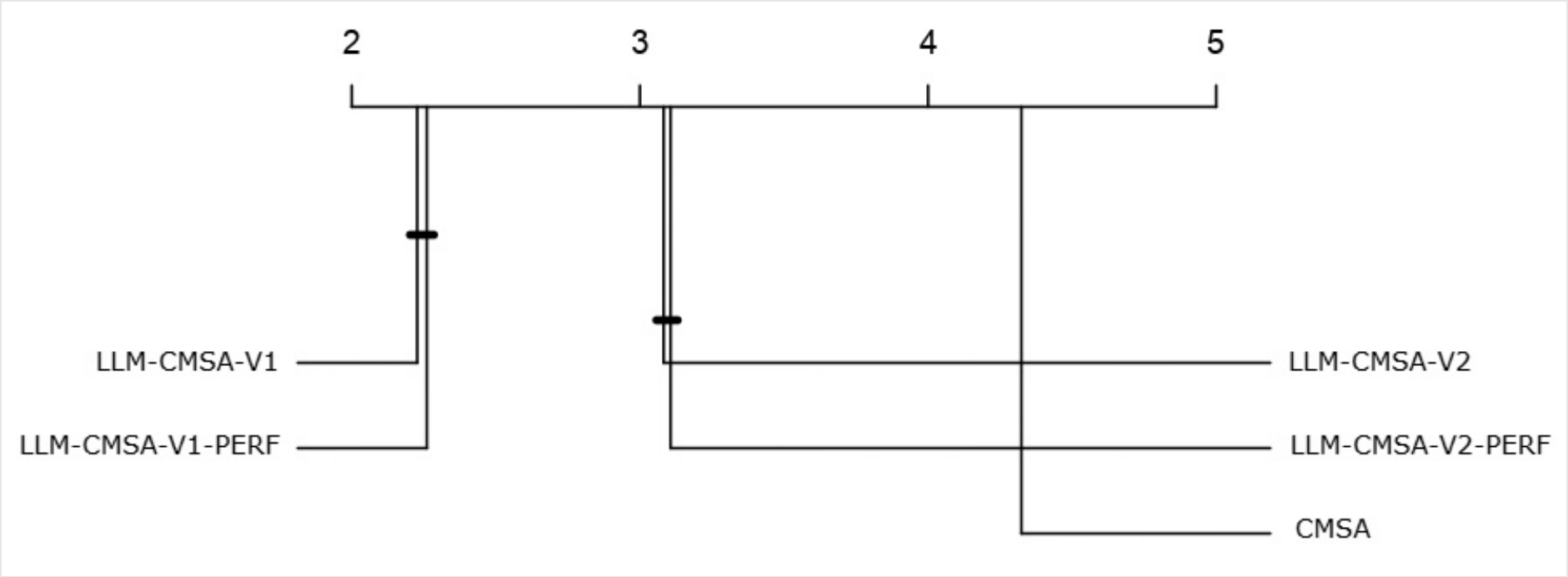






Results: Erdös-Rényi Graphs

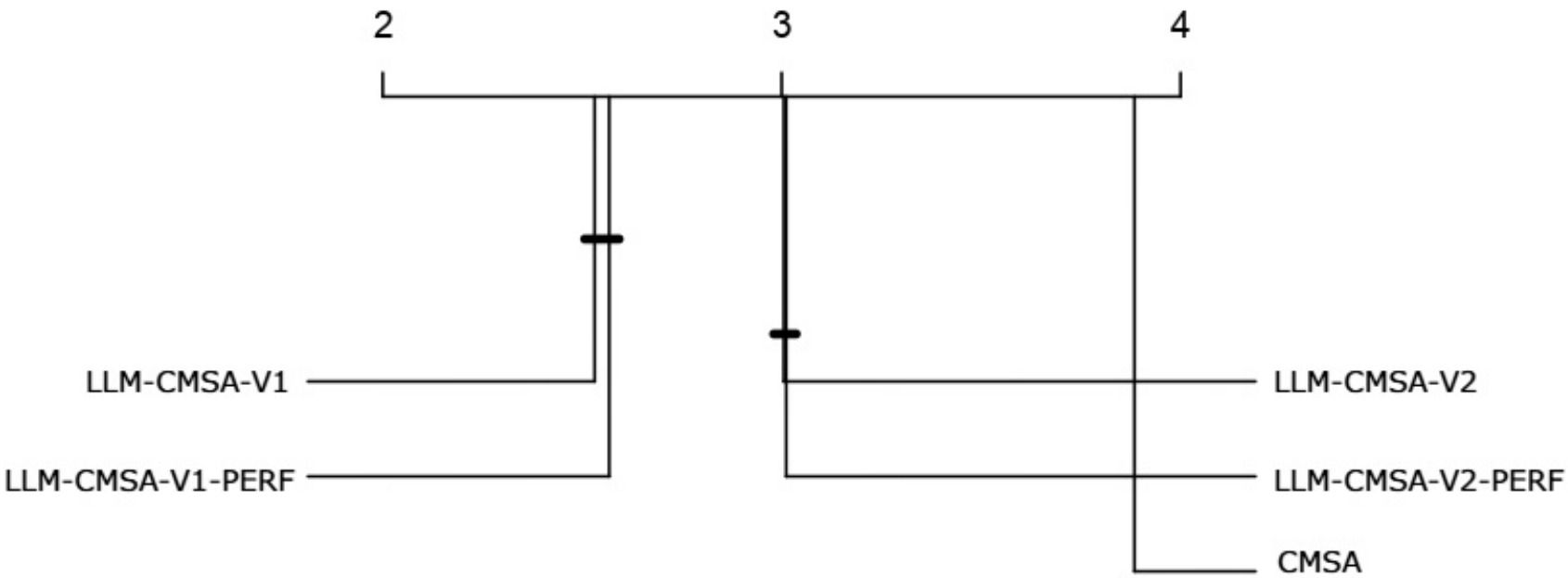
Critical Difference Plot



Note: Critical Difference plots are a visual tool used in statistical comparisons of multiple algorithms (or models) across multiple datasets. They are commonly used in machine learning and statistics to display the results of non-parametric multiple comparison tests, such as the Nemenyi test, after performing a Friedman test.

Results: Watts-Strogatz Graphs

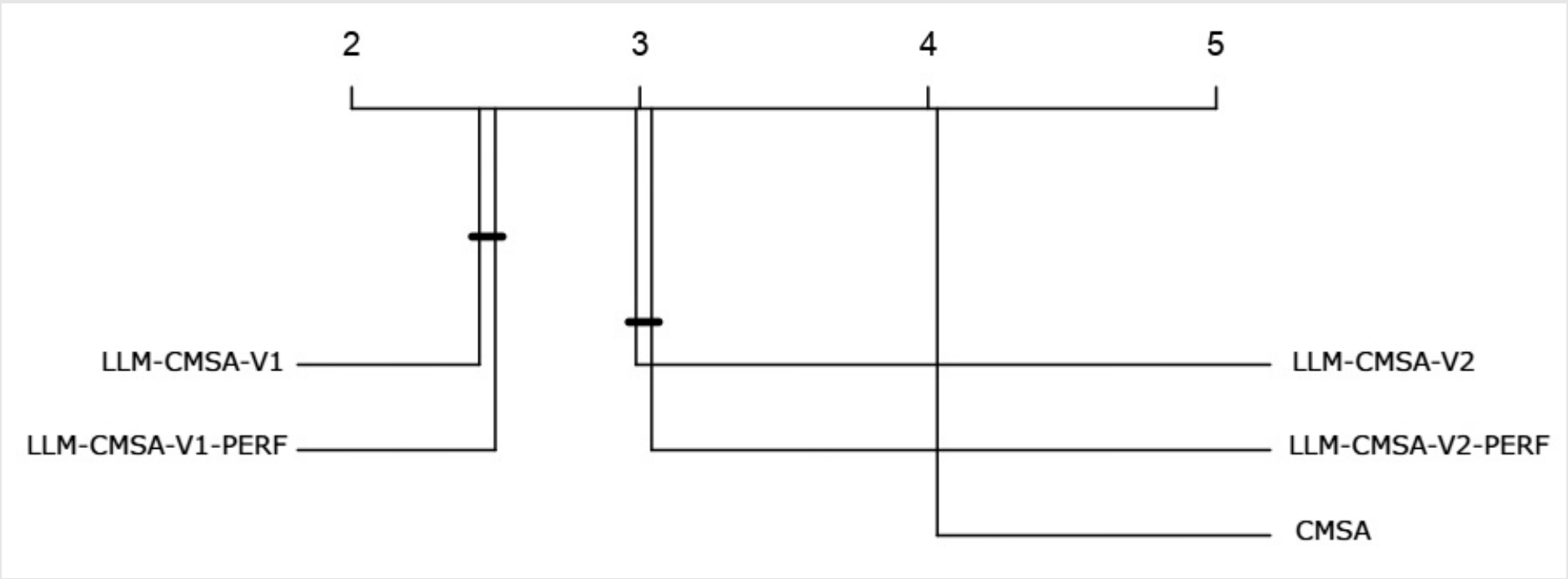
Critical Difference Plot



Note: Critical Difference plots are a visual tool used in statistical comparisons of multiple algorithms (or models) across multiple datasets. They are commonly used in machine learning and statistics to display the results of non-parametric multiple comparison tests, such as the Nemenyi test, after performing a Friedman test.

Results: Barabási-Albert Graphs

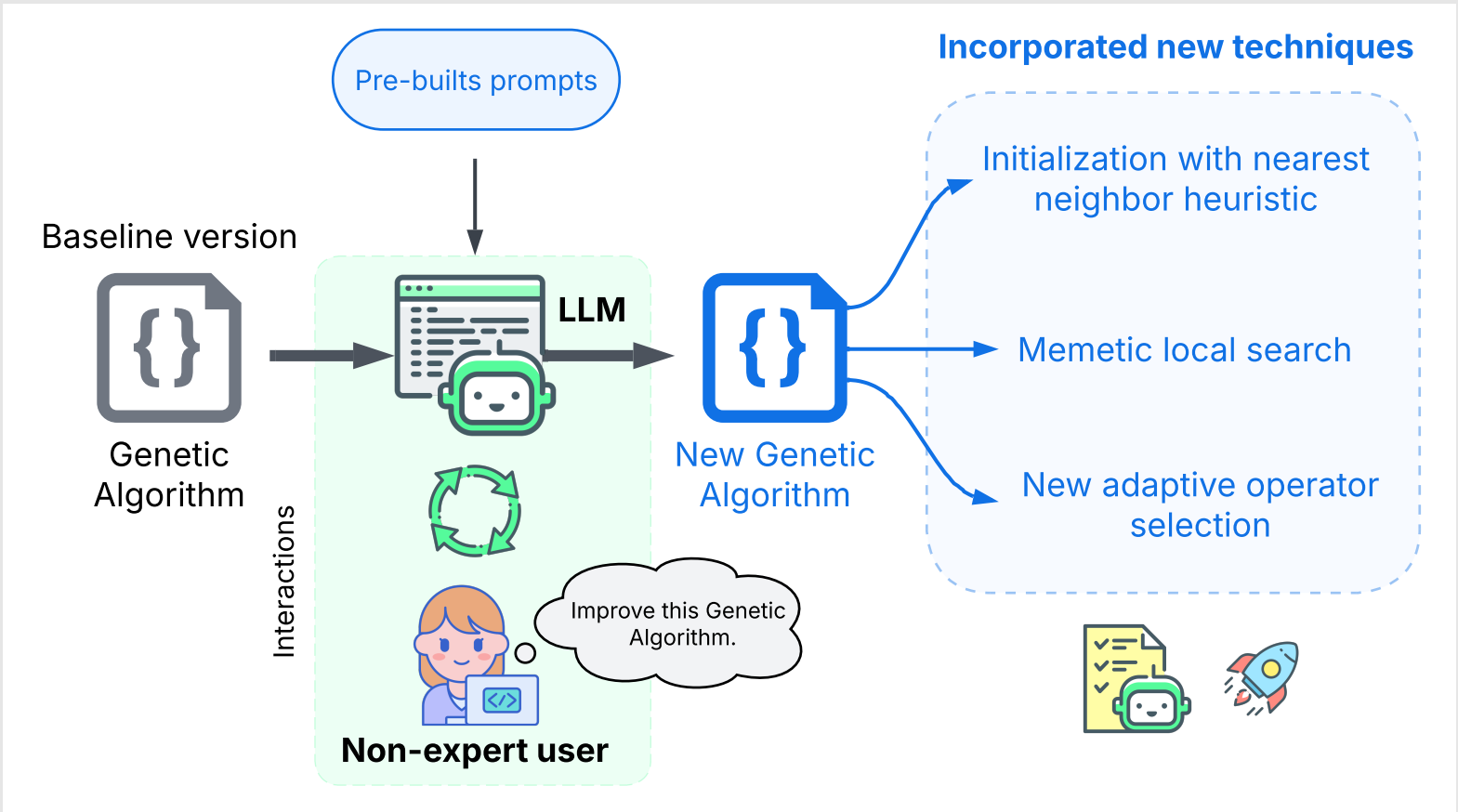
Critical Difference Plot



Note: Critical Difference plots are a visual tool used in statistical comparisons of multiple algorithms (or models) across multiple datasets. They are commonly used in machine learning and statistics to display the results of non-parametric multiple comparison tests, such as the Nemenyi test, after performing a Friedman test.

More General Research Line

Improve Optimization Code



https://doi.org/10.48550/arXiv.2503.10968

Under journal review

Experiments: Chosen Problem

Example Problem: TSP

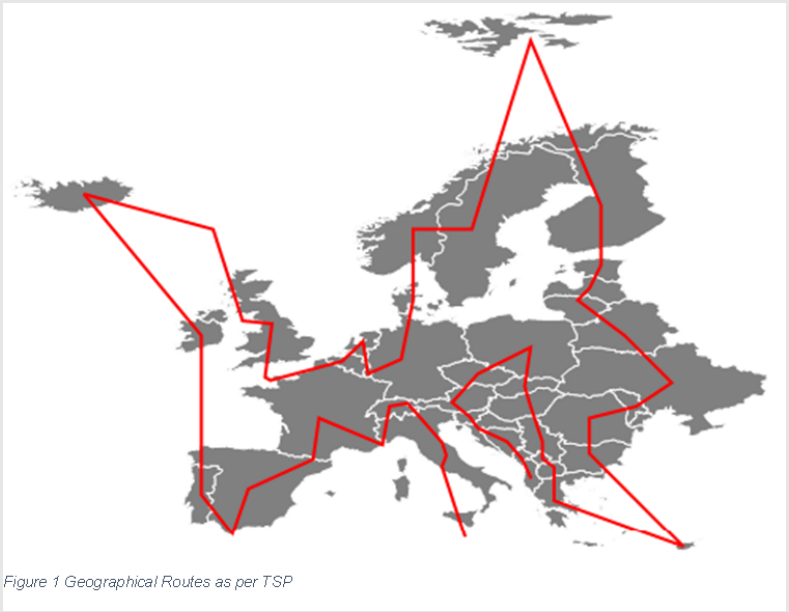


Figure 1 Geographical Routes as per TSP

Advantages

- Emblematic Combinatorial Optimization Problem
- There is a good chance LLMs know about the TSP
- Open source code of algorithms readily available

Chosen Algorithms and LLMs

Python Library pyCombinatorial: Chosen Algorithms

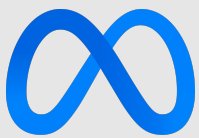
- Ant Colony Optimization
- Genetic Algorithm
- Adaptive Large Neighborhood Search
- Tabu Search
- Simulated Annealing
- Q-Learning (Reinforcement Learning)
- SARSA (Reinforcement Learning)
- Christofides heuristic
- Convex hull heuristic
- Branch & Bound

URL: https://github.com/Valdecy/pyCombinatorial

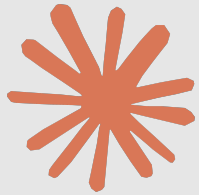
Chosen LLMs

OpenAI's GPT

Google's Gemini



Meta's LLaMA



Anthropic's Claude

DeepSeek

Utilized Prompt Template

Prompt Template

```
You are an optimization algorithm expert.

I need to improve this {{algorithm
      name}} implementation for the travelling salesman problem (TSP)
      by incorporating state-of-the-art techniques. Focus on:

1. Finding better quality solutions
2. Faster convergence time

Requirements:
- Keep the main function signature: {{the signature of an the main
      function}}
- Include detailed docstrings explaining:
  * What improvement is implemented
  * How it enhances performance
  * Which state-of-the-art technique it is based on
- All explanations must be within docstrings, no additional text
- Check that there are no errors in the code

IMPORTANT:
- Return ONLY Python code
- Any explanation or discussion must be inside docstrings
- At the end, include a comment block listing unmodified functions from
      the original code

Current implementation:
{{algorithm code}}
```

Generating LLM-Updated Codes

Possible Code Generation Errors

- Code causes compilation/execution errors
- Code compiles and runs fine, but produces invalid TSP solutions

Success in Code Generation

Algorithm	Claude-3.5-Sonnet (temp = 1.0)		Gemini-exp-1206 (temp = 2.0)		Llama-3.3-70B (temp = 1.0)		GPT-O1 (temp = 1.0)		DeepSeek-R1 (temp = 1.0)	
	Success		Success		Success		Success		Success	
	1st Try	# Attempts	1st Try	# Attempts	1st Try	# Attempts	1st Try	# Attempts	1st Try	# Attempts
ACO	✗	1	✓	-	✓	-	✓	-	✓	-
GA	✓	-	✗	1	✓	-	✗✗	3	✓	-
ALNS	✗	1	✓	-	✓	-	✓	-	✗✗	3
TABU	✓	-	✓	-	✓	-	✓	-	✓	-
SA	✓	-	✓	-	✓	-	✓	-	✓	-
Q_Learning	✗	-	✓	-	✓	-	✓	-	✓	-
SARSA	✗	-	✓	-	✓	-	✓	-	✓	-
Christofides	✗	-	✓	-	✓	-	✓	-	✓	-
Convex Hull	✗	1	✓	-	✓	-	✓	-	✓	-
Branch and Bound	✓	-	✗✗	4	✓	-	✓	-	✓	-

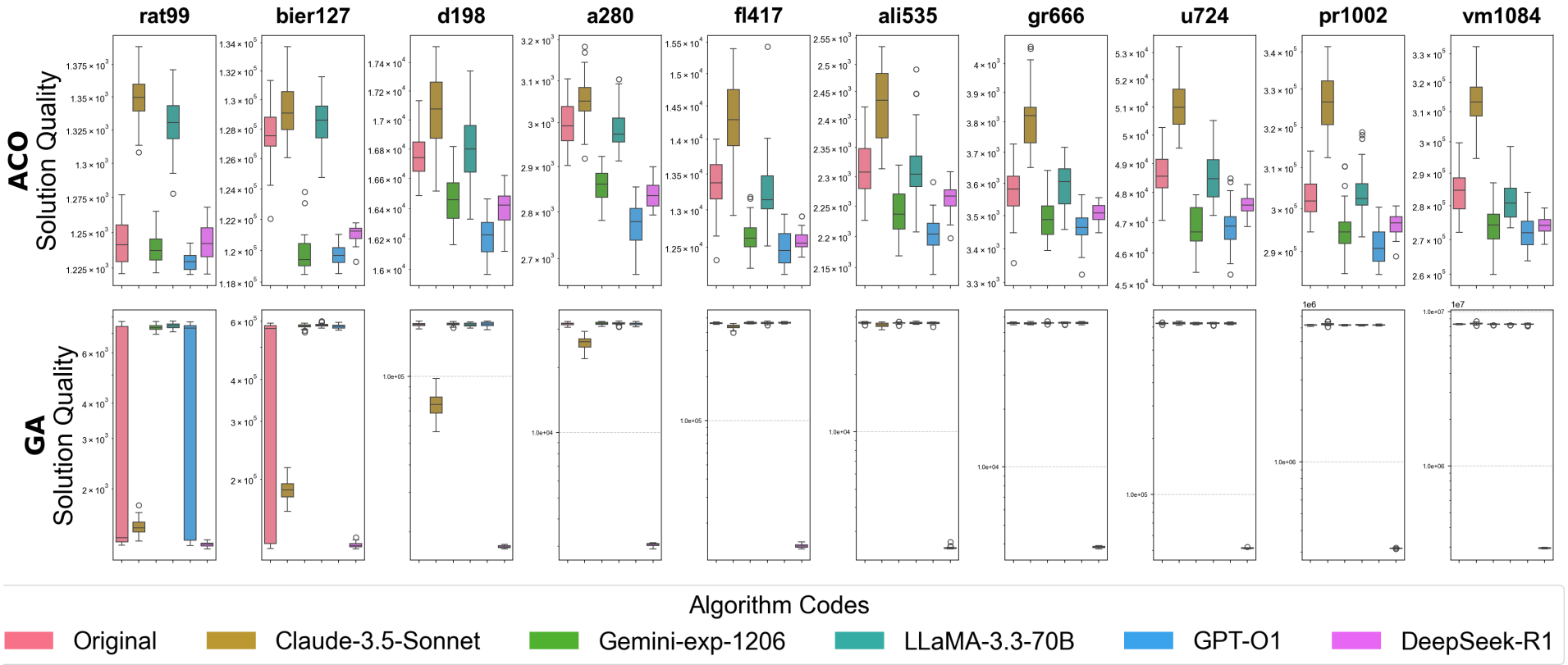
Experimental Setup

Problem Instances, Algorithms, and Tuning

- **10 instances:** **rat99**, **bier127**, **d198**, **a280**, **fl417**, **ali535**, **gr666**, **u724**, **pr1002**, **vm1084**
- **Algorithm codes:** 60 different ones ((Original + (5*LLM)) * 10 algorithms)
- **Algorithm tuning:** all 60 codes are tuned with the scientific tuning software `irace`[a]

[a] López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives, 3, 43-58.

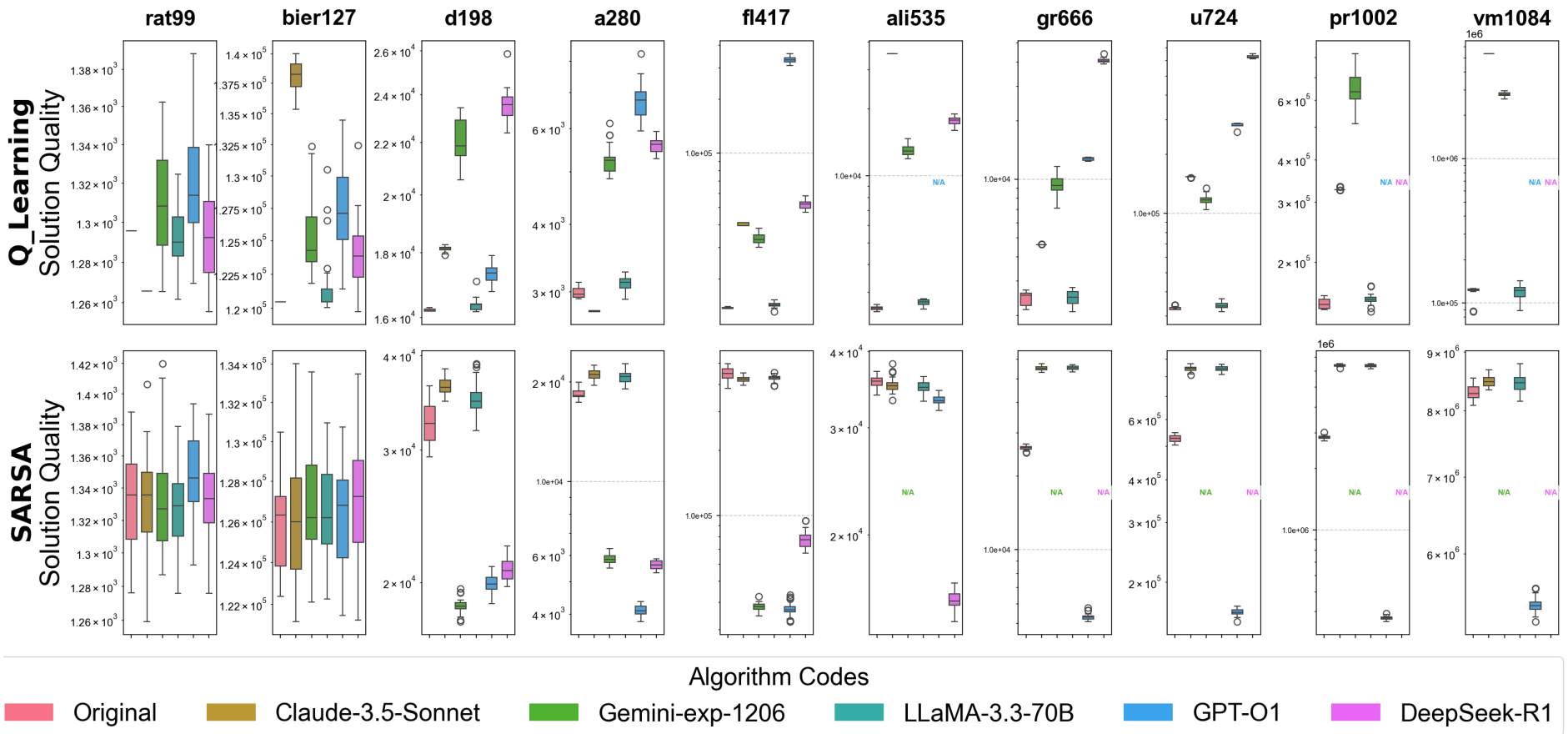
Results: ACO and GA



Examples of Introduced Improvements

- DeepSeek: heuristic initialization of the initial GA population
- GPT: local pheromone update to make other ants explore new paths

Results: Reinforcement Learning

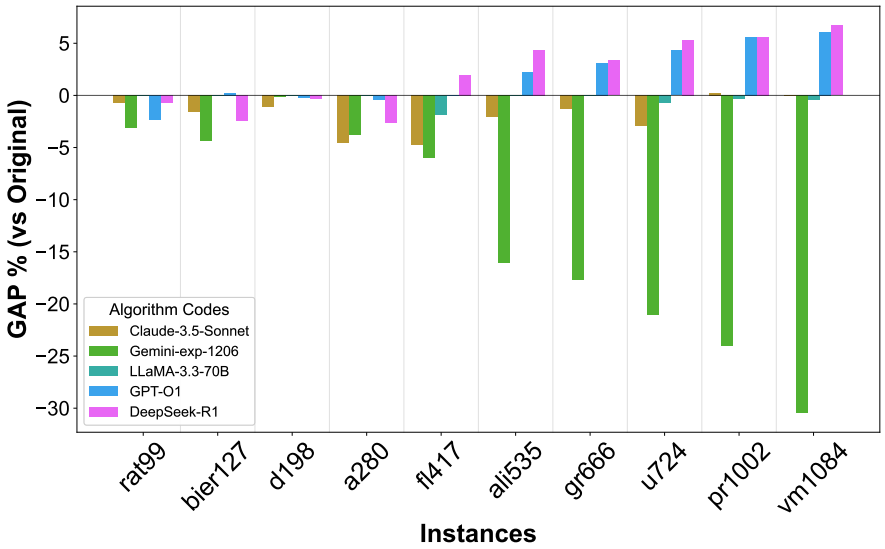


Example of Introduced Improvements in SARSA

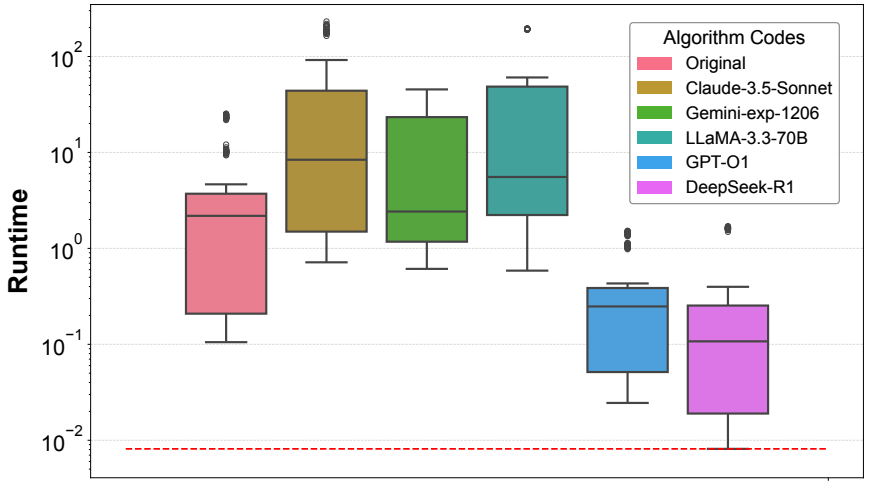
- GPT: makes use of Boltzmann Exploration for solution construction

Results: Christofides and B&B

Christofides



Branch & Bound



Example improvement

GPT: Uses an enhanced 2-opt local search with the "Don't Look Bits" technique.

Example improvement

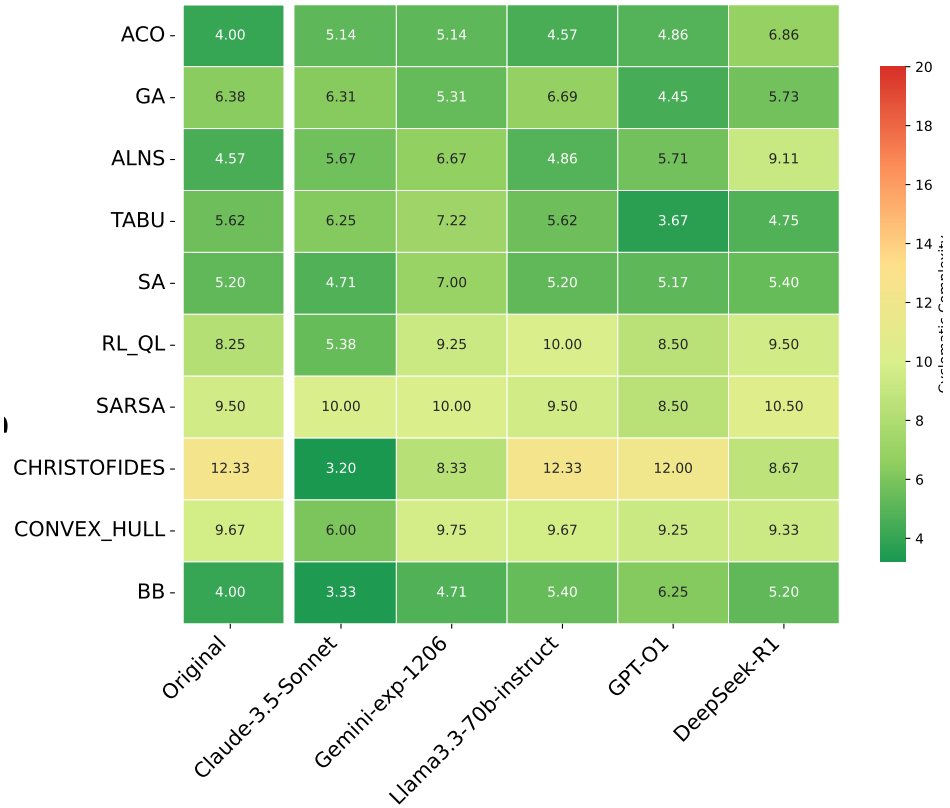
DeepSeek: Initialization with a heuristic solution and dynamical sorting of candidates

Results: Code Complexity

Cyclomatic Complexity

A metric that quantifies the number of independent paths through a program's source code[a]

[a] Ebert, C., Cain, J., Antoniol, G., Counsell, S., & Laplante, P. (2016). Cyclomatic complexity. IEEE software, 33(6), 27-29.



Observations

- Average complexity score of best-performing models is 6.84 (GPT) and 7.51 (DeepSeek), which is considered low and indicates well-structured code.
- Better and less complex than original: GPT SARSA code, DeepSeek GA and Christofides codes

Results: Summary

- GPT and DeepSeek generally produce the best results
- Gemini performed well in certain cases (ACO and ALNS); however, it underperformed in others, such as Christofides
- Claude presented the lowest performance.
- In 9 out of 10 cases (algorithms), LLMs were able to come up with improved code

LLMs: Challenges and Concerns

- **Bias and Fairness**: Models can reflect societal biases in training data.
- **Misinformation**: Can generate plausible but incorrect information.
- **Energy Consumption**: Training large models consumes substantial resources.
- **Security and Misuse**: Risk of harmful outputs or malicious use.

Questions?

Drop me an Email: christian.blum@iiia.csic.es