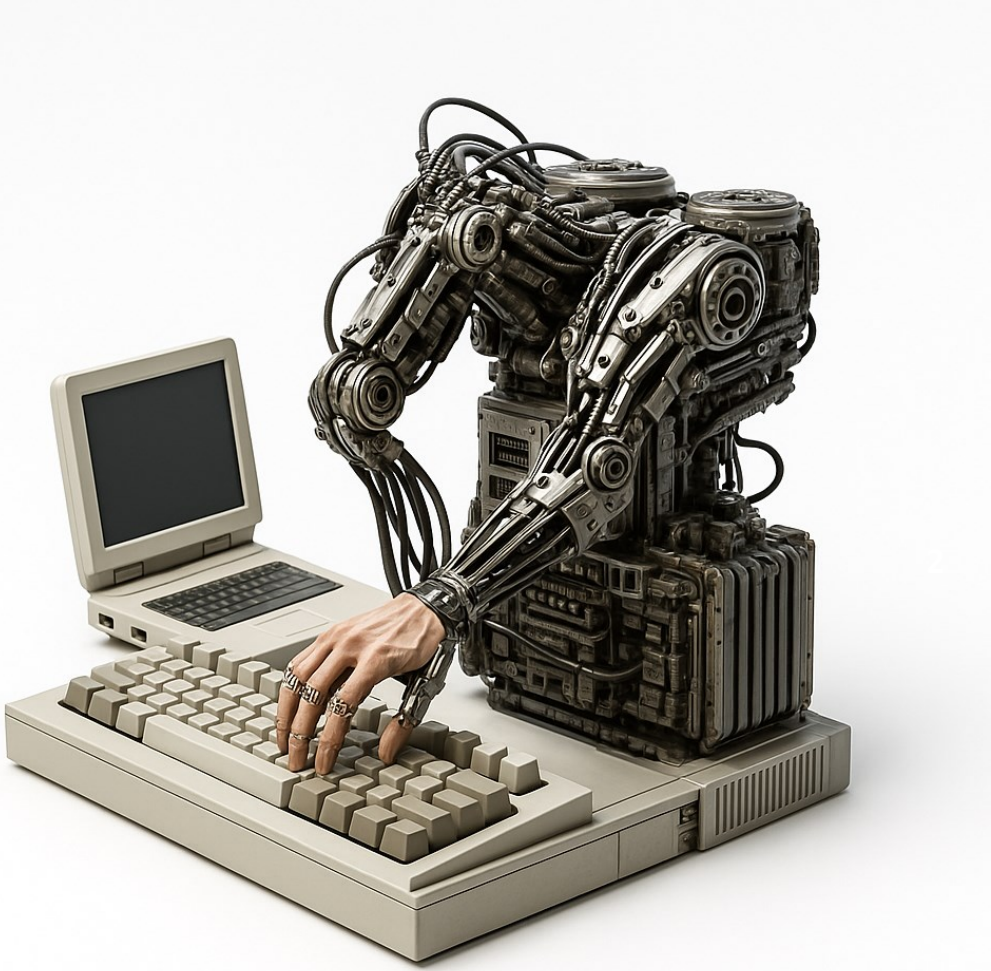
AGILE WORKFLOWS & GENERATIVE AI in A New Era of Autonomous Optimization

Seyedali Mirjalili
Professor
May 2025

VŠB TECHNICKÁ UNIVERZITA OSTRAVA
TORRENS UNIVERSITY AUSTRALIA

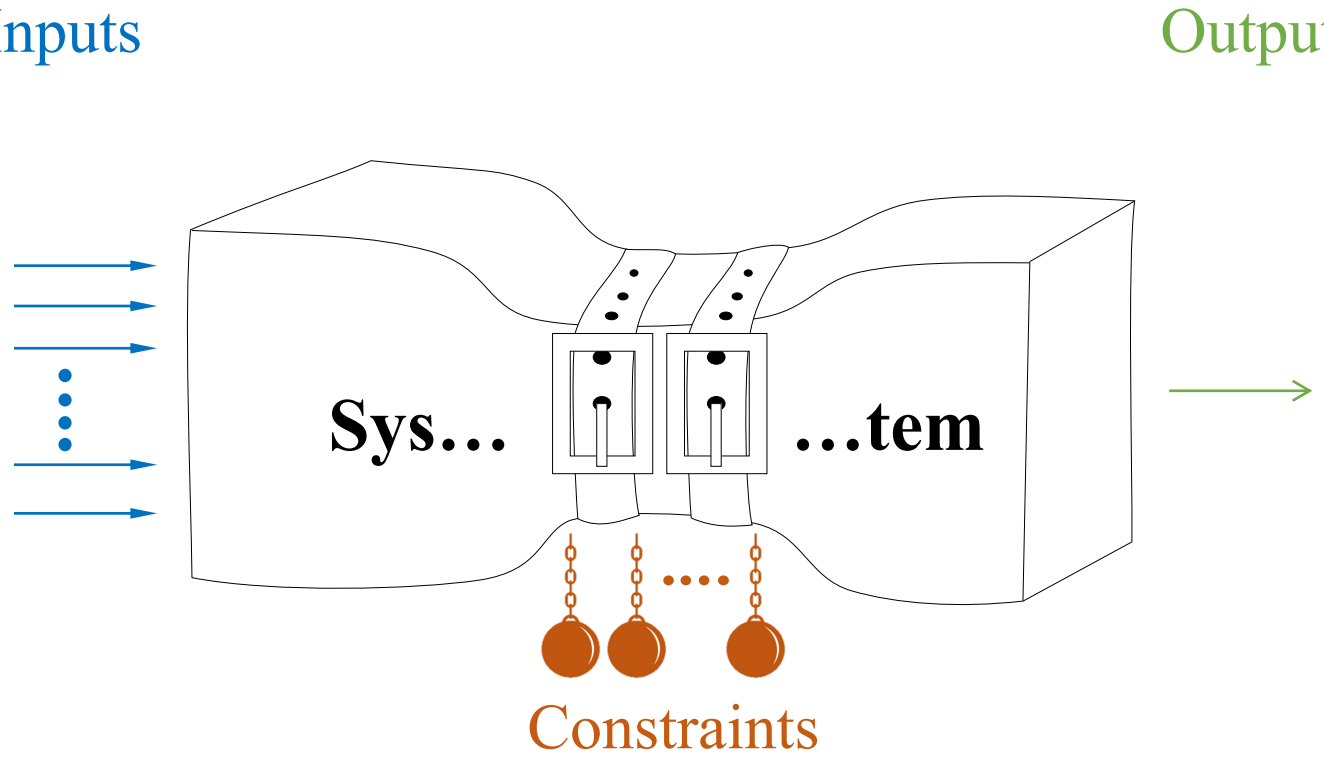
CONTENTS



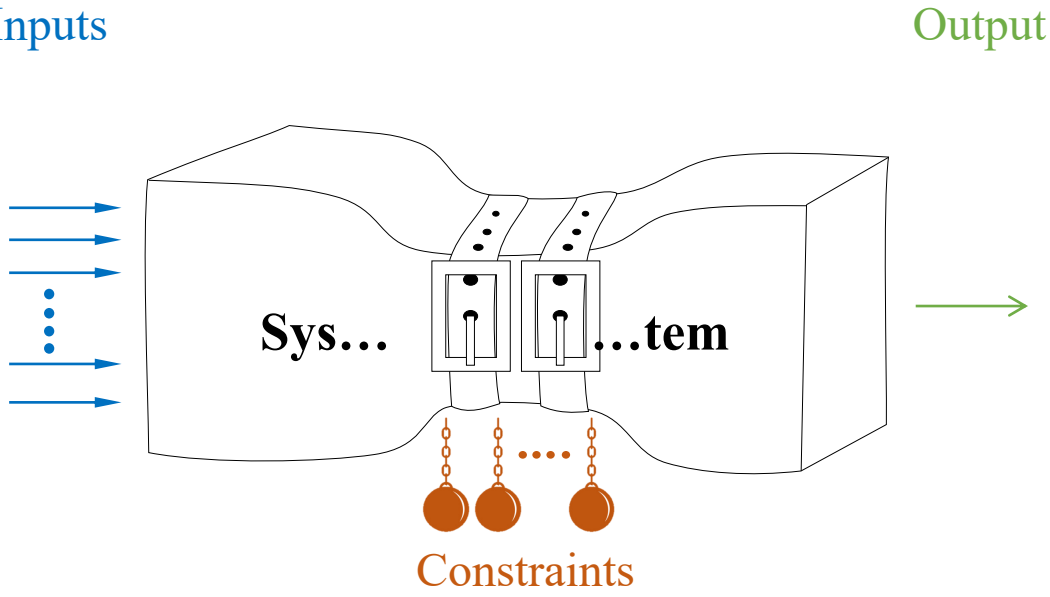
AI-generated Image

OPTIMIZATION PROBLEMS

MAIN COMPONENTS OF AN OPTIMIZATION PROBLEM

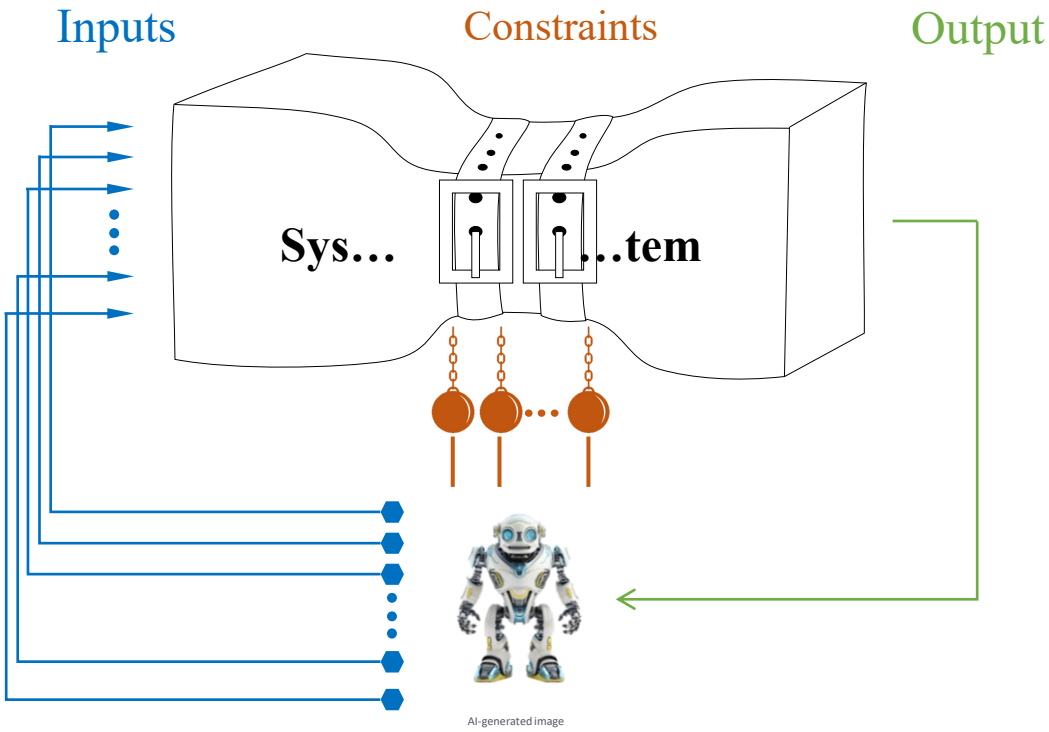


FORMULATION OF AN OPTIMIZATION PROBLEM

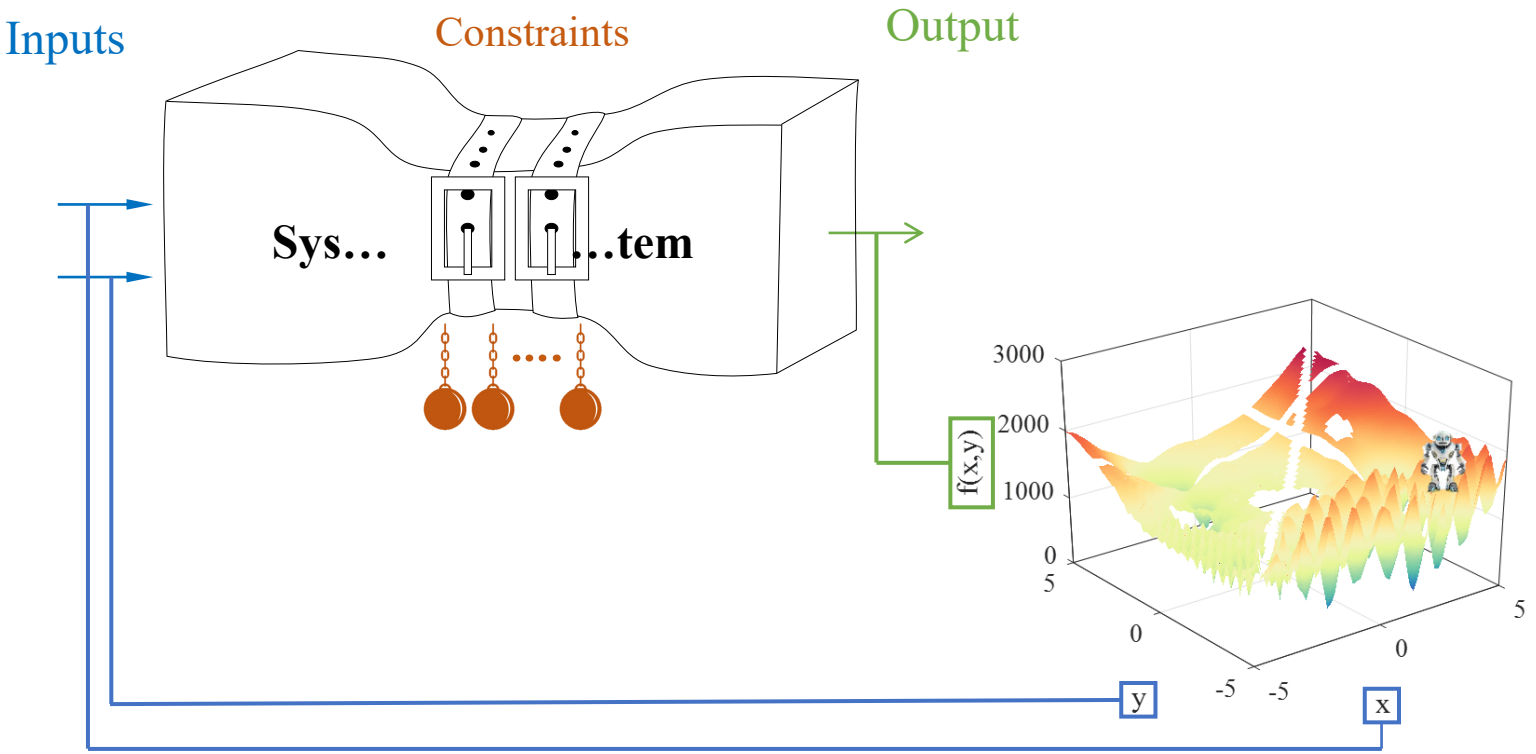


$$Minimise: \quad f(x_1, x_2, \ldots, x_n)$$
$$Suject\ to: \quad Constraints$$

THE ROLE OF AN OPTIMIZATION ALGORITHM

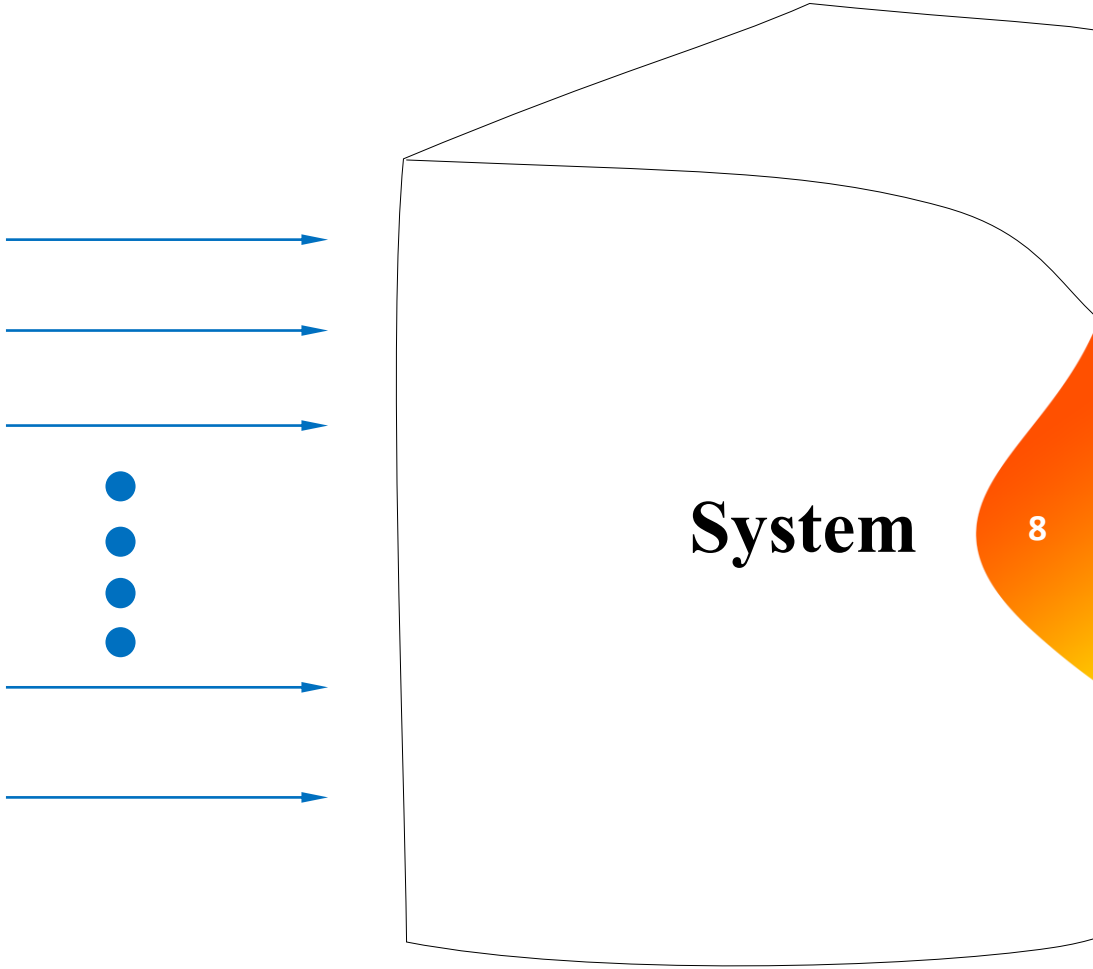


WHAT ARE WE ACTUALLY TRYING TO DO?

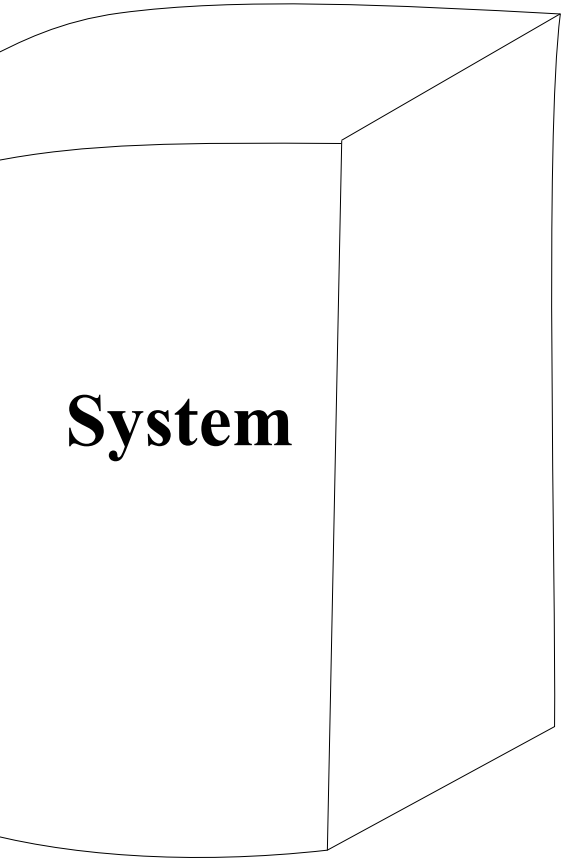


INPUTS

- Large number of inputs
- Variables with different ranges
- Dependency between the inputs
- Discrete variables
- Mix variables
- Noisy inputs
- ….



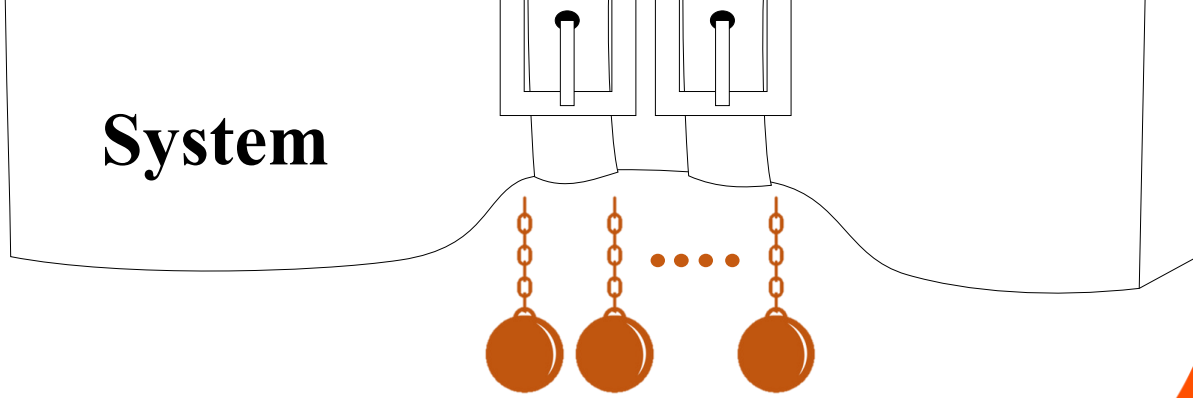
OUTPUTS



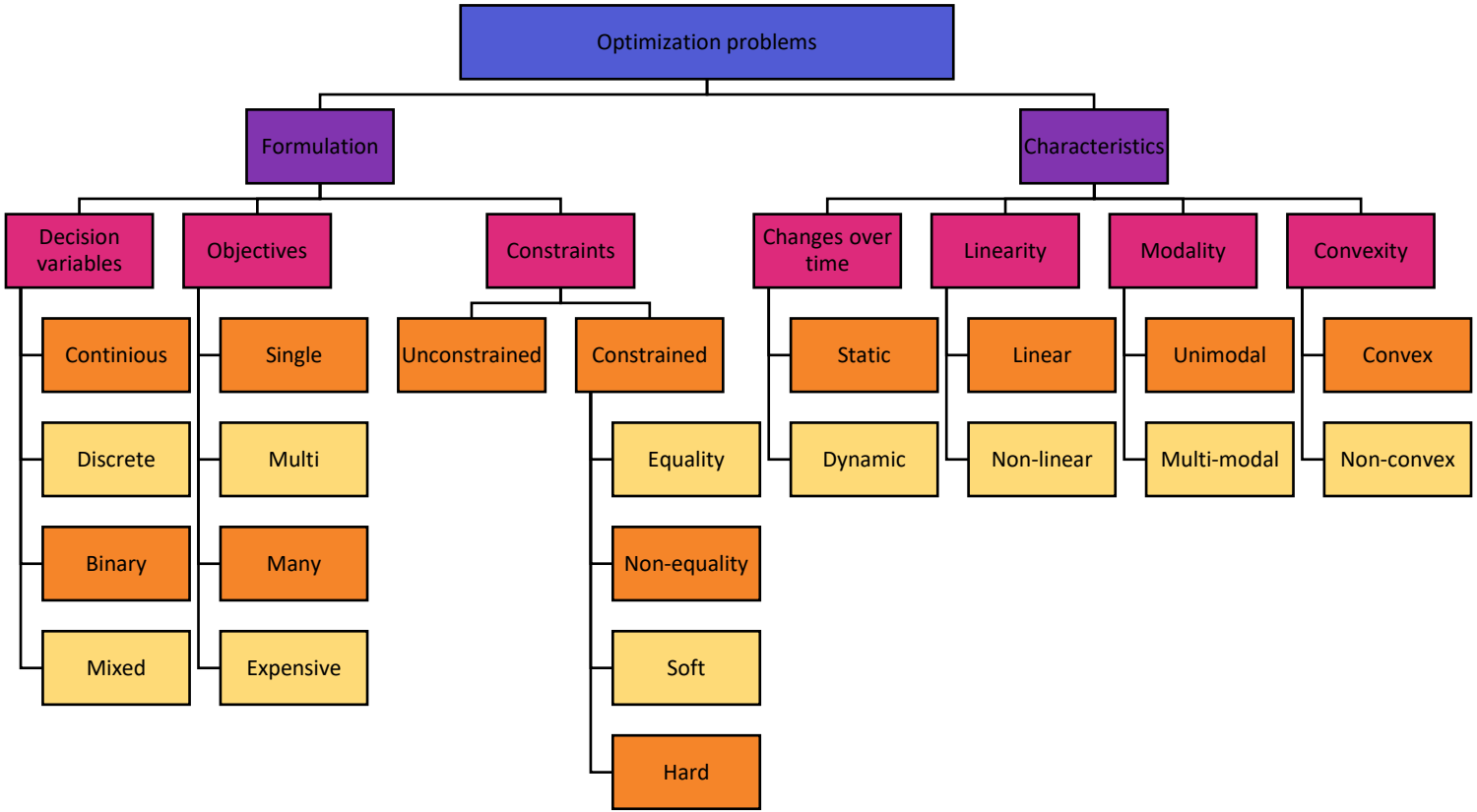
- Multiple/many objectives
- Conflicting objectives
- Dynamic objectives
- Difficult-to-Measure Objectives
- …

CONSTRAINTS

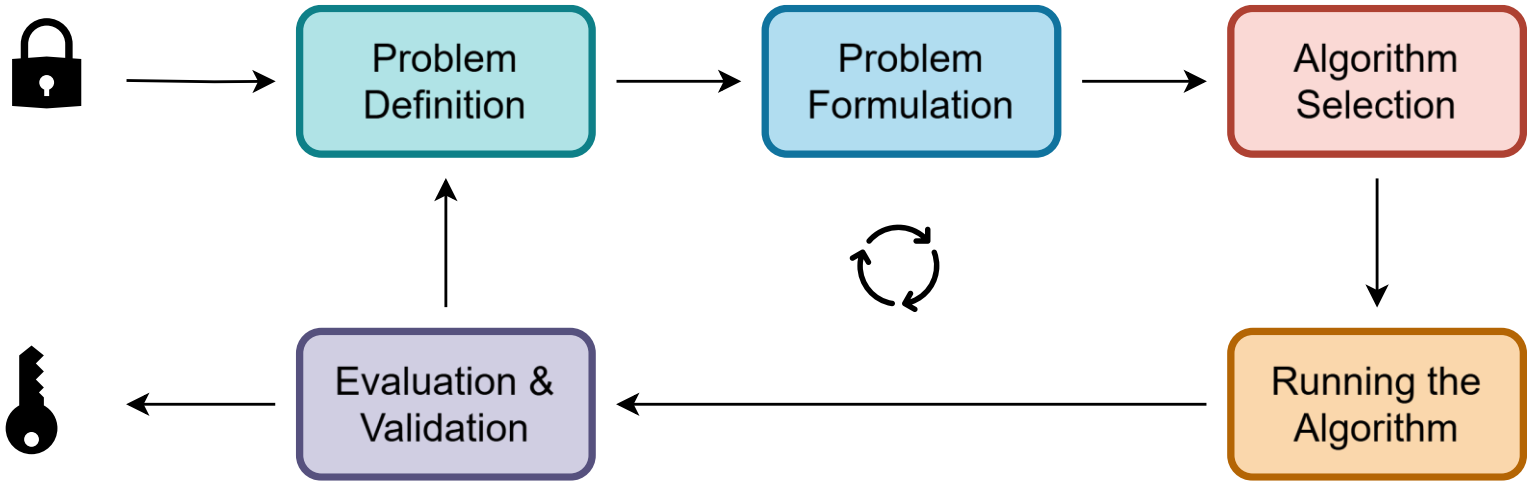
- Highly constrained landscape
- Equality constraint
- Inequality constraint
- Priority (soft vs. hard)
- Non-linear constraints
- Dynamic constraints
- …



A TAXONOMY

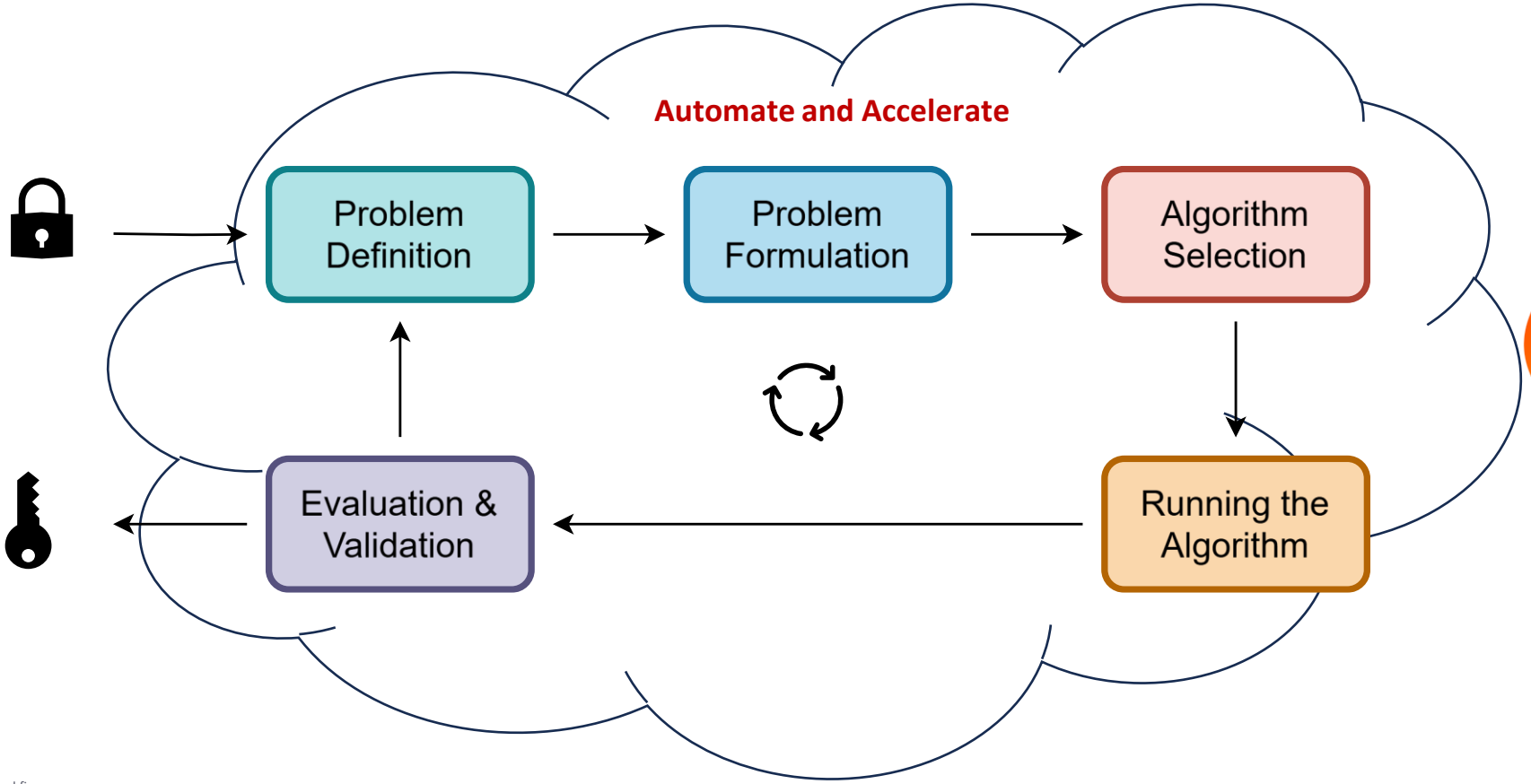

Optimization problems
Formulation
Characteristics
Decision variables
Objectives
Constraints
Changes over time
Linearity
Modality
Convexity
Continious
Discrete
Binary
Mixed
Single
Multi
Many
Expensive
Unconstrained
Constrained
Equality
Non-equality
Soft
Hard
Static
Dynamic
Linear
Non-linear
Unimodal
Multi-modal
Convex
Non-convex

OPTIMIZATION LIFECYCLE



Human-generated figure

OPTIMIZATION LIFECYCLE

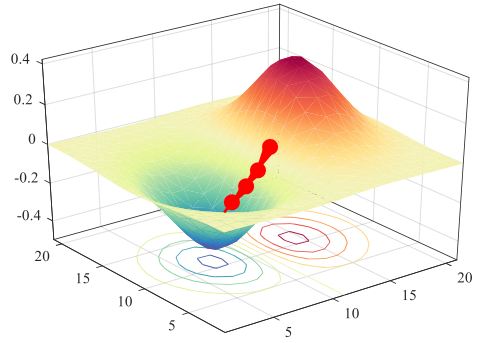


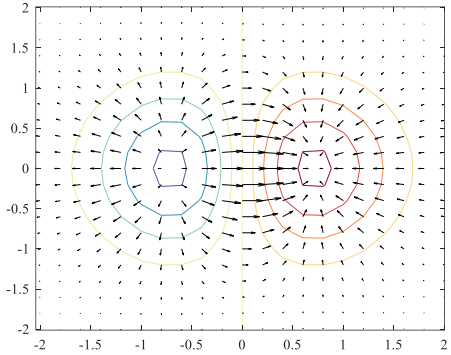
Human-generated figure

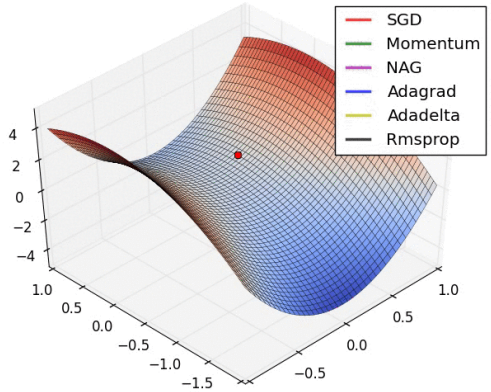
INCREASING AUTOMATION

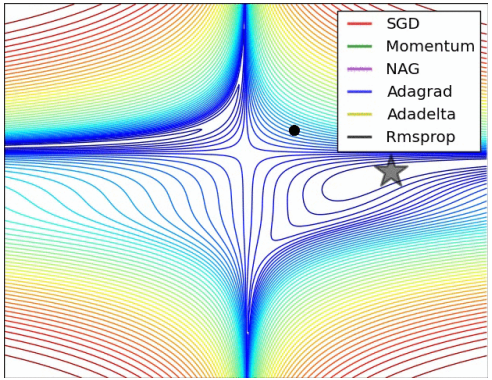
CONVENTIONAL ALGORITHMS

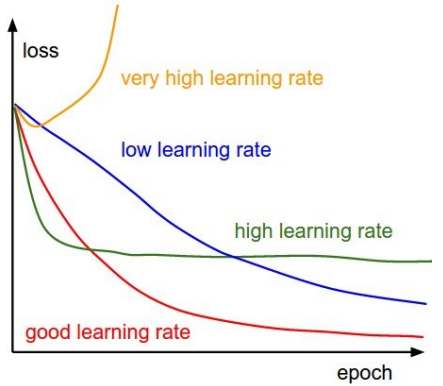
1. Most Gradient-based
2. Sensitive to learning rate
3. Local optima
4. Saddle points
5. Not very practical



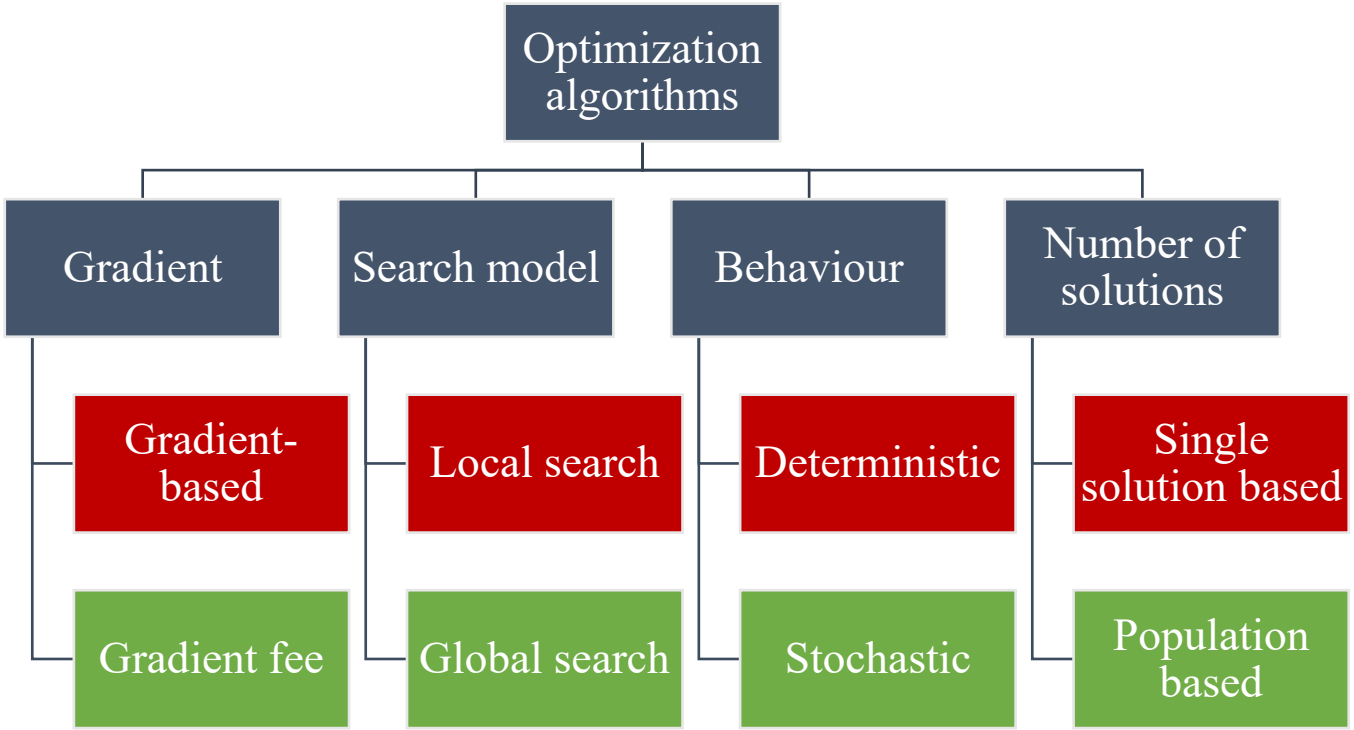




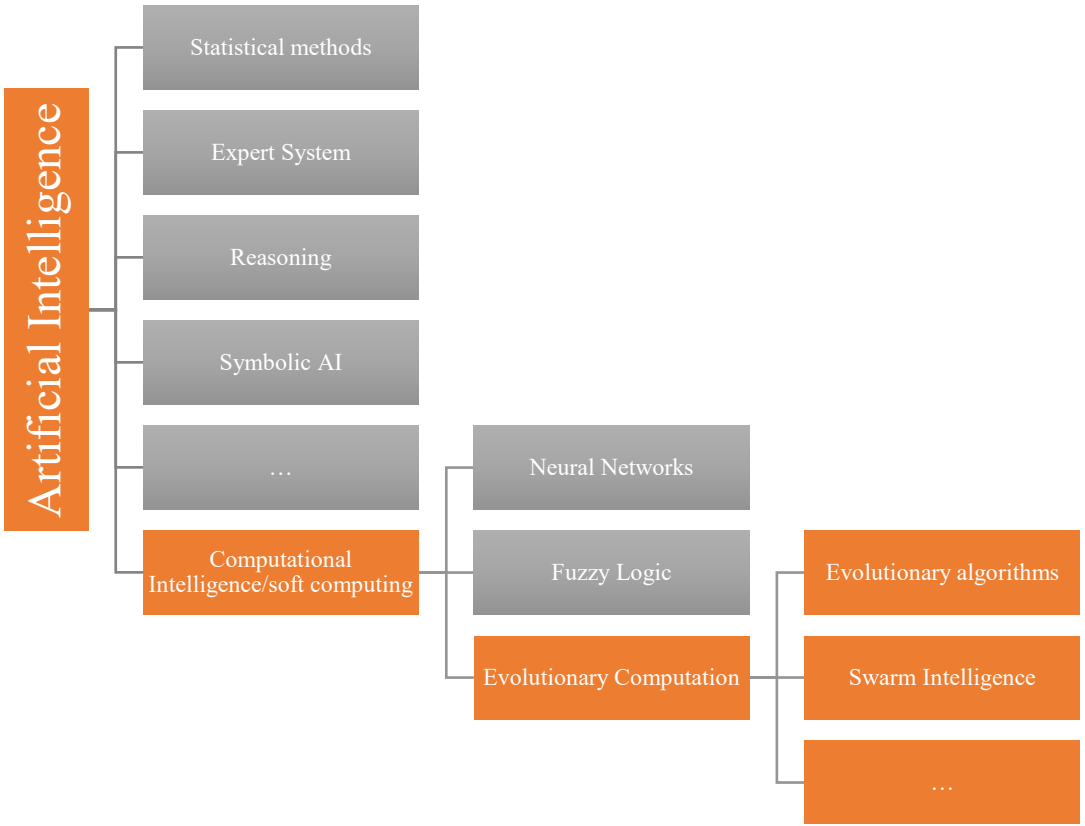




RECENT ALGORITHMS



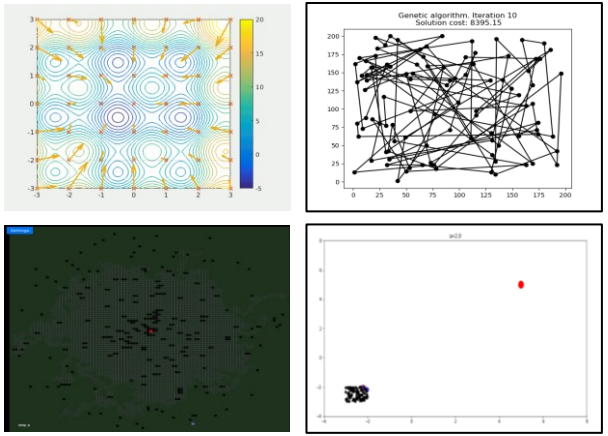
EVOLUTIONARY COMPUTATION & SOFT COMPUTING



MODERN ALGORITHMS

Advantages:

1. Avoid local solutions
2. Higher chance of finding the global optimum
3. Low dependency on the initial solution
4. Mostly do not need gradient

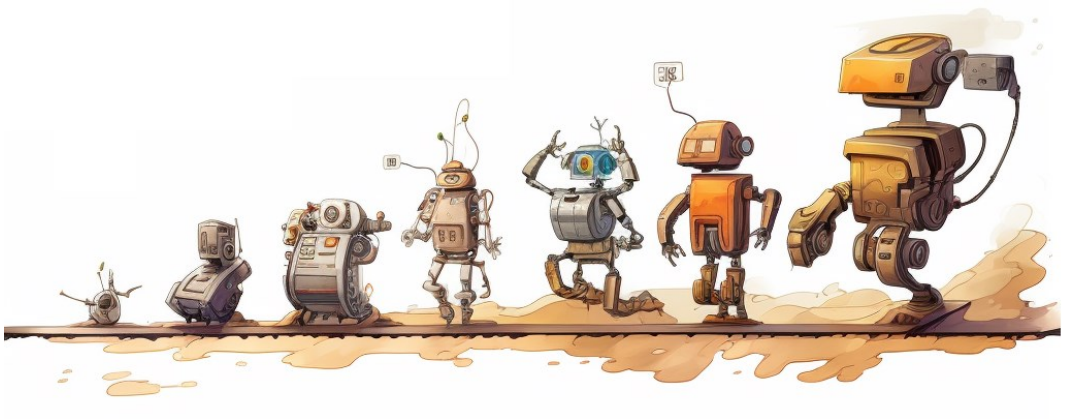


Drawbacks:

1. Slow convergence speed
2. Finding different answers in each run

WHERE WER ARE HEADING?

Gradient-based → Heuristic → Meta-heuristic → Hyper-heuristic → ?????

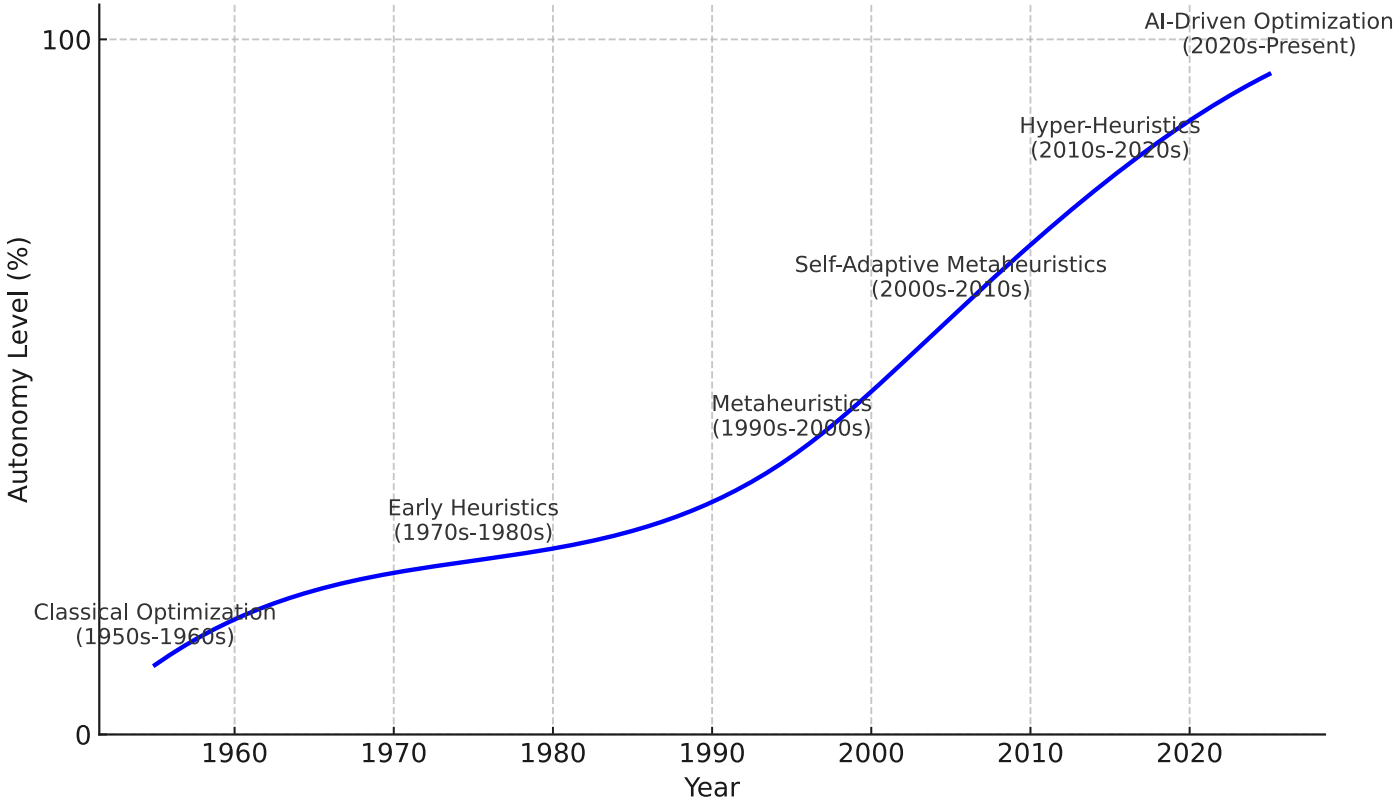


AI-generated image

Increased Automation

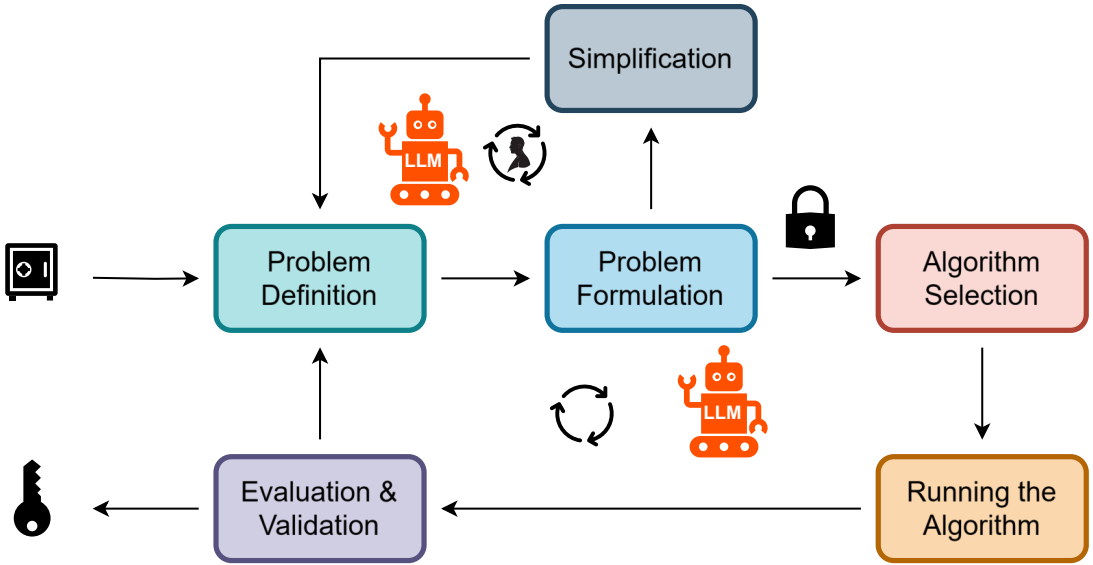
Less Human Involvement

WHERE WER ARE HEADING?



OPEN RESEARCH QUESTIONS

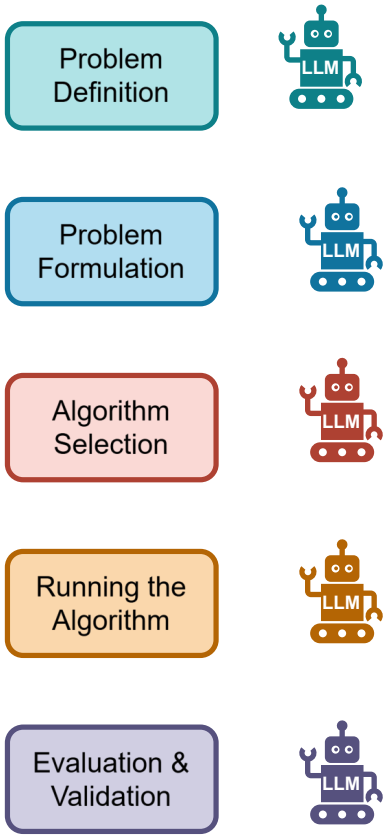
1. How to automate algorithm development and improvement?
2. How to automate problem formulation?
3. How to reduce human involvement in both?



Human-generated figure

HOW GENAI CAN BE USED?

1. GenAI-powered problem definition:
   - Gathers and structures qualitative data for precise requirement modeling.
   - Maximizes stakeholder input and validates requirements for optimization.
2. GenAI-powered problem formulation:
   - Creates accurate mathematical models from detailed requirements.
   - Refines models iteratively to match real-world conditions effectively.
3. GenAI-powered algorithm design and development:
   - Generates, analyzes, and refines code for algorithm development.
   - Enables rapid exploration of innovative meta-heuristic solutions.
4. GenAI-powered algorithm executor:
   - Optimizes hardware and algorithm settings for efficient execution.
   - Monitors and improves execution by addressing inefficiencies.
5. GenAI-powered solution evaluator:
   - Evaluates results, identifies patterns, and suggests improvements.
   - Accelerates iterative optimization with expert-level insights.



HOW GENAI CAN BE IN ALGORITHM DESIGN AND DEVELOPMENT?

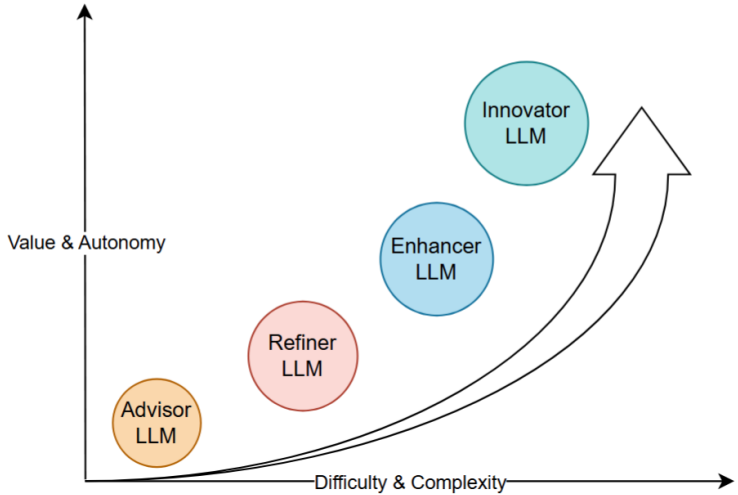
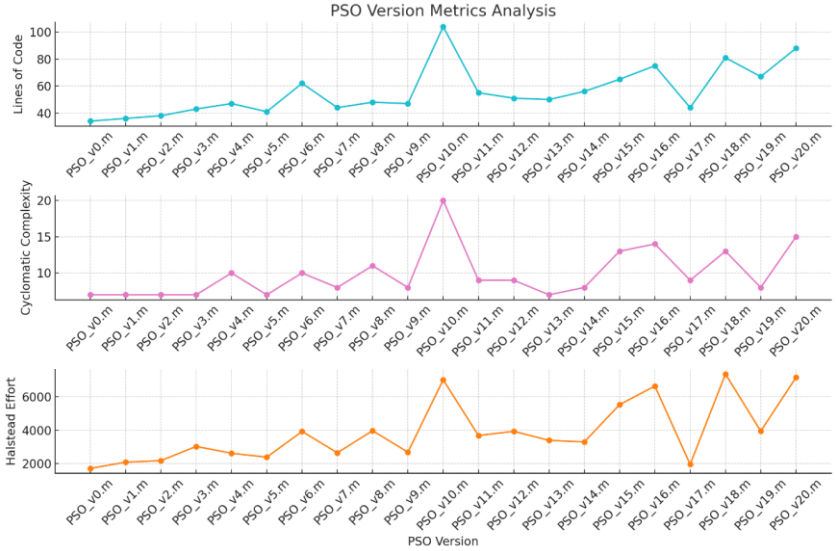


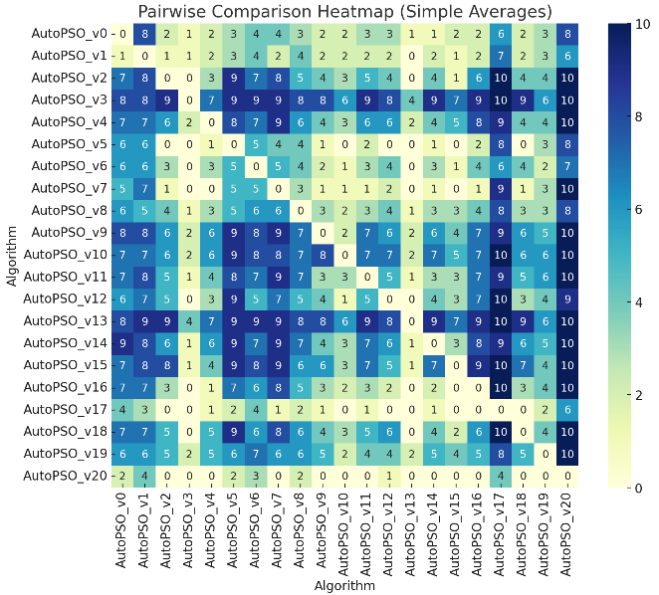
Figure 14: Visualisation of LLM Roles in metaheuristic optimisation where Advisor, Refiner, Enhancer, and Innovator roles each contribute uniquely to improving algorithm performance and adaptability.

EXAMPLE 1

Algorithm Development

Improving PSO 20 times without human inevrtion using GPT 4o (temperature 0.7)





EXAMPLE 2

Problem Formulation | Algorithm Development | LLM

Sustainable Facility Site Location Selection

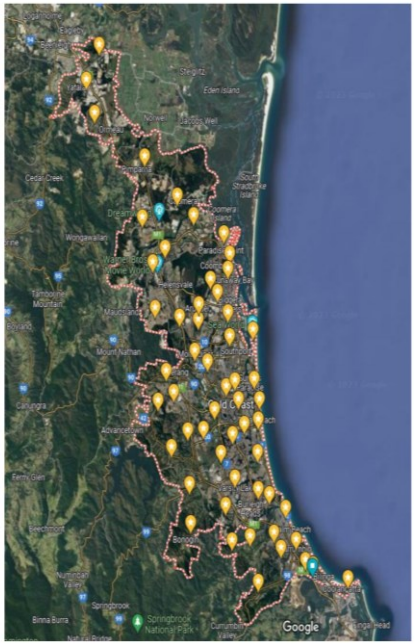


Figure 6. All DCs available for Gold Coast City (source Google Maps)[32]

Choose a set of distribution-centre locations that minimises the total distance travelled (a proxy for overall transport cost) while satisfying geographic, capacity and service-level constraints.

Why it is challenging

- The search space grows exponentially with the number of candidate sites (NP-hard).
- Multiple, often conflicting criteria (cost, coverage, environmental impact, risk) must be balanced.
- Real-world data are noisy and dynamic (demand shifts, new constraints).

EXAMPLE 2

Problem Formulation

Algorithm Development

Sustainable Facility Site Location Selection

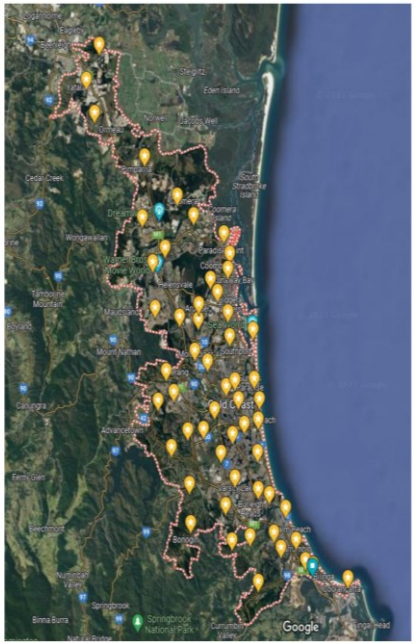
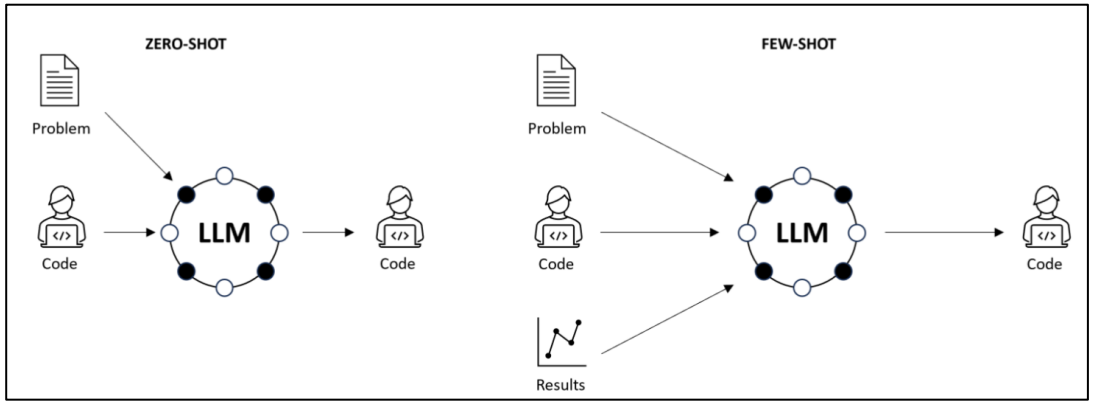


Figure 6. All DCs available for Gold Coast City (source Google Maps)[32]



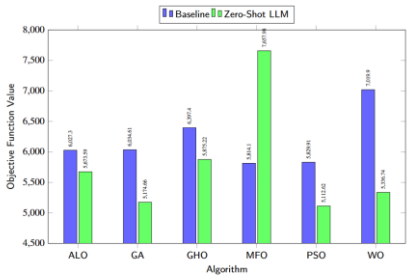


Figure 17: Performance comparison of baseline and zero-shot LLM-enhanced algorithms under Configuration B

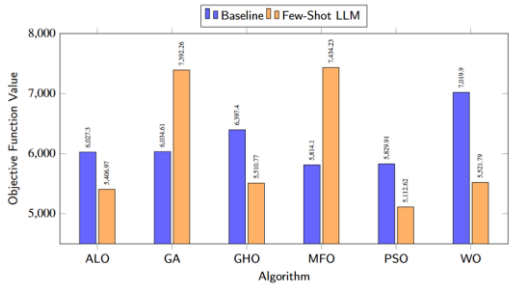


Figure 25: Performance comparison of baseline and few-shot LLM-enhanced algorithms under Configuration B

EXAMPLE 2

Problem Formulation

Algorithm Development

LLM

Sustainable Facility Site Location Selection

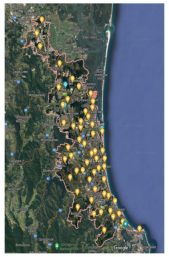


Figure 6. All DCs available for Gold Coast City (source Google Maps)[32]

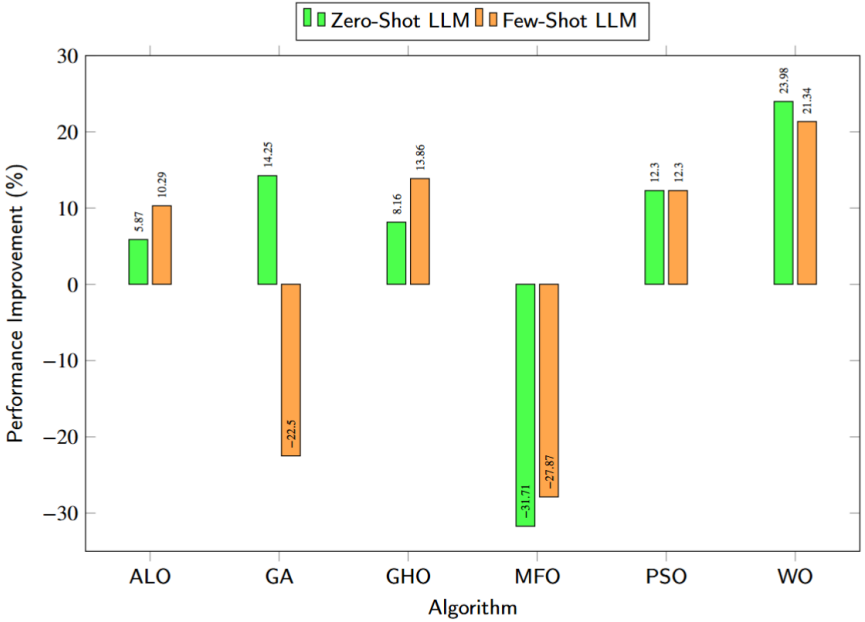


Figure 33: Comparative performance improvement across enhancement approaches (%)

EXAMPLE 2

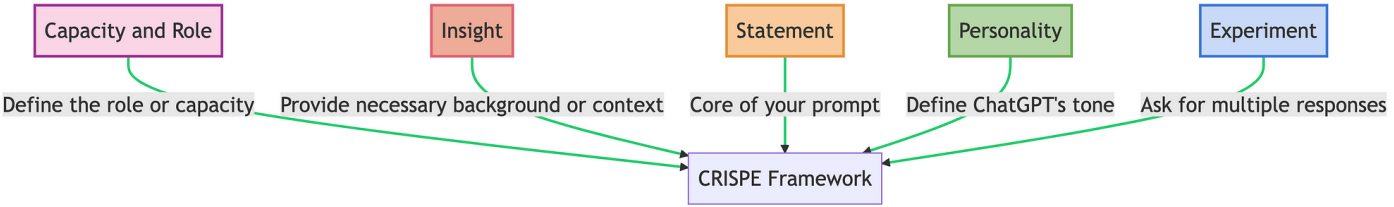
Problem Formulation

Algorithm Development

LLM

We used a generic prompt engineering framework

Figure 6. All DCs available for Gold Coast City (source Google Maps)[32]



EXAMPLE 3: A NEW PROMPTING FRAMEWORK

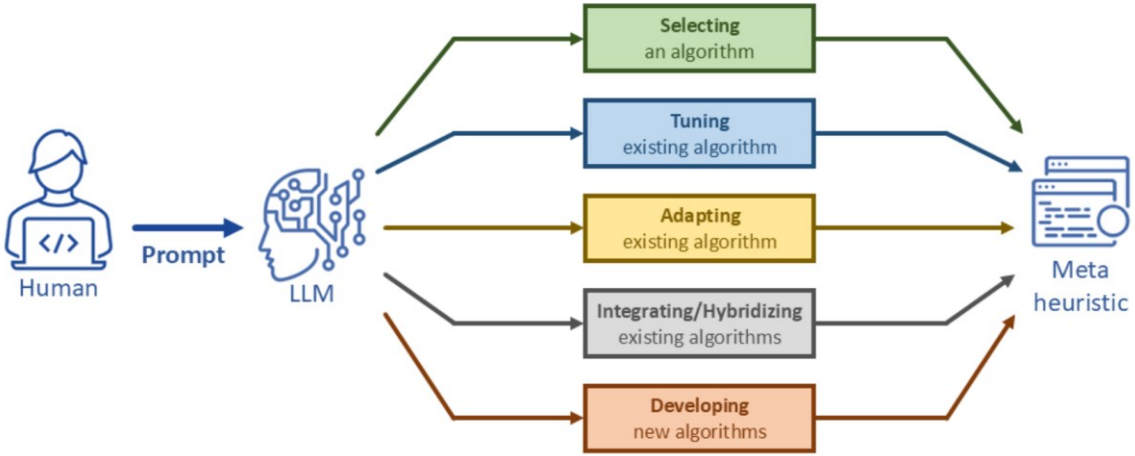


Figure 2: Human and LLM Collaboration in Meta-Heuristic Optimization Across Five Levels: Selection Level, Tuning Level, Adapting Level, Integration Level, and Developing Level

EXAMPLE 3: A NEW PROMPTING FRAMEWORK

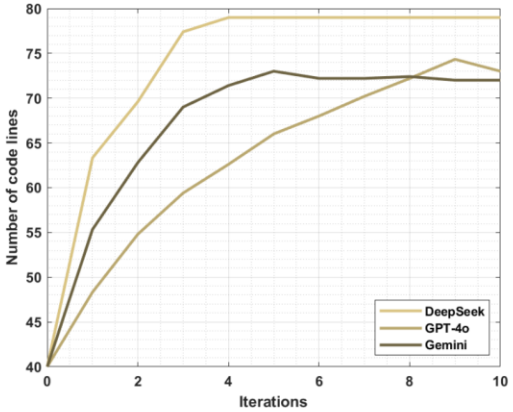
Table 2
RESOLUTION Hyper-Framework Aspects Across Levels

RESOLUTION	Aspects	Selection Level	Tuning Level	Adapting Level	Integration Level	Developing Level
R: **Role**	Explaining roles	*	*	*	*	*
E: **Explanation**	Explaining what we require	*	*	*	*	*
S: **inStances**	Defining the optimization problem instance	*	*	*	*	*
O: **algOrithms**	Introducing the algorithm(s)	*	*	*	*	*
L: **variabLes**	Determining variables to be tuned	-	*	*	*	*
U: **modUles**	Introducing the pool of modules	-	-	*	*	*
T: **deTails**	Determining details of the desired output code	*	*	*	*	*
I: **archItecture**	Determining structure or architecture of the output code	*	*	*	*	*
O: **scOre**	Defining the scoring criteria	*	*	*	*	*
N: **returN**	Explaining what we require as the output, including algorithm(s), experimental confirmation, final code(s), etc.	*	*	*	*	*

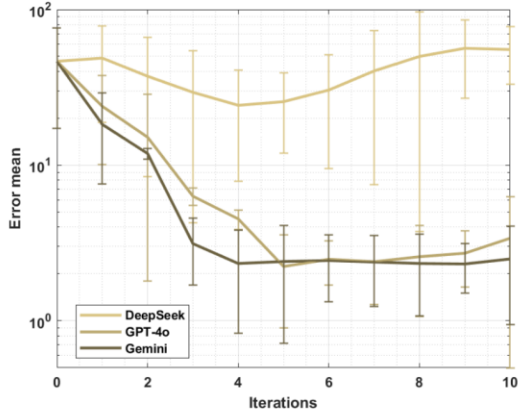
EXAMPLE 3: A NEW PROMPTING FRAMEWORK

RESOLUTION Prompt (Adapting Existing Algorithms for Solving Traveling Salesman Problem)

Consider the following as an input prompt and do so carefully to the end. **R**oles phase (You act as a code developer. You act as an optimization expert.) **E**xplanation phase (I am looking for an optimized binary GA code based on the provided code to use in my project.) In**S**tances phase(To solve a TSP problem instance with the given distance matrix.) Alg**O**rithms phase (I would like you to precisely adjust the parameters of the code and the operators used in the code.) Variab**L**es phase (Consider tuning all parameters and variables.) Mod**U**les phase (Consider all possible operators could be found in MH literature.) De**T**ails phase (I require a detailed Python code, with extensive comments to explain each part.) Arch**I**tecture phase (Revising the structure of the code is also allowed.) Sc**O**re phase (Revise the original code to enhance it based on a metric defined as the product of the answer error and the number of lines of the code) Retur**N** phase (1- Provide the revised code. 2- Highlight any tuning made. 3- Run both the original and the revised codes for 10 times to solve the instance with the given distance matrix. 4- Which algorithm do you suggest based on the mentioned score average? 5- Provide a table of the number of lines and the tour distance mean and standard deviation for both codes confirming your answer.)

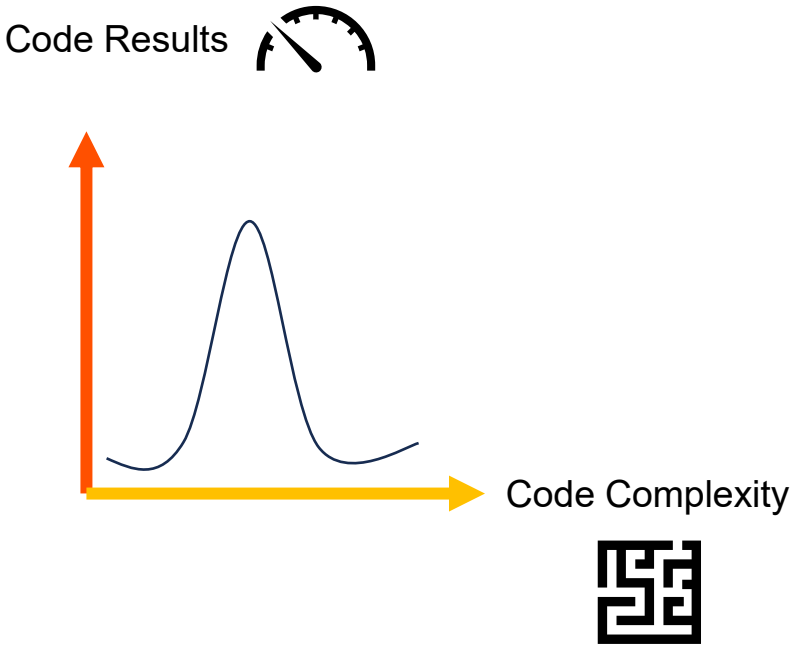


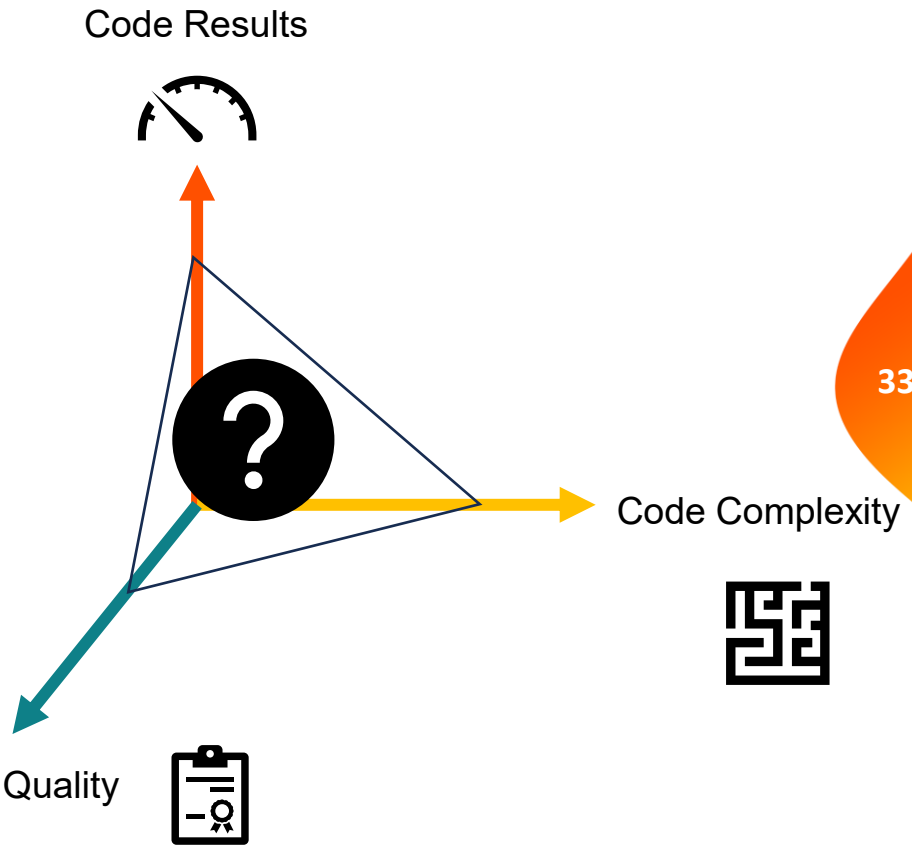
(a) Number of code lines.



(b) Error mean.

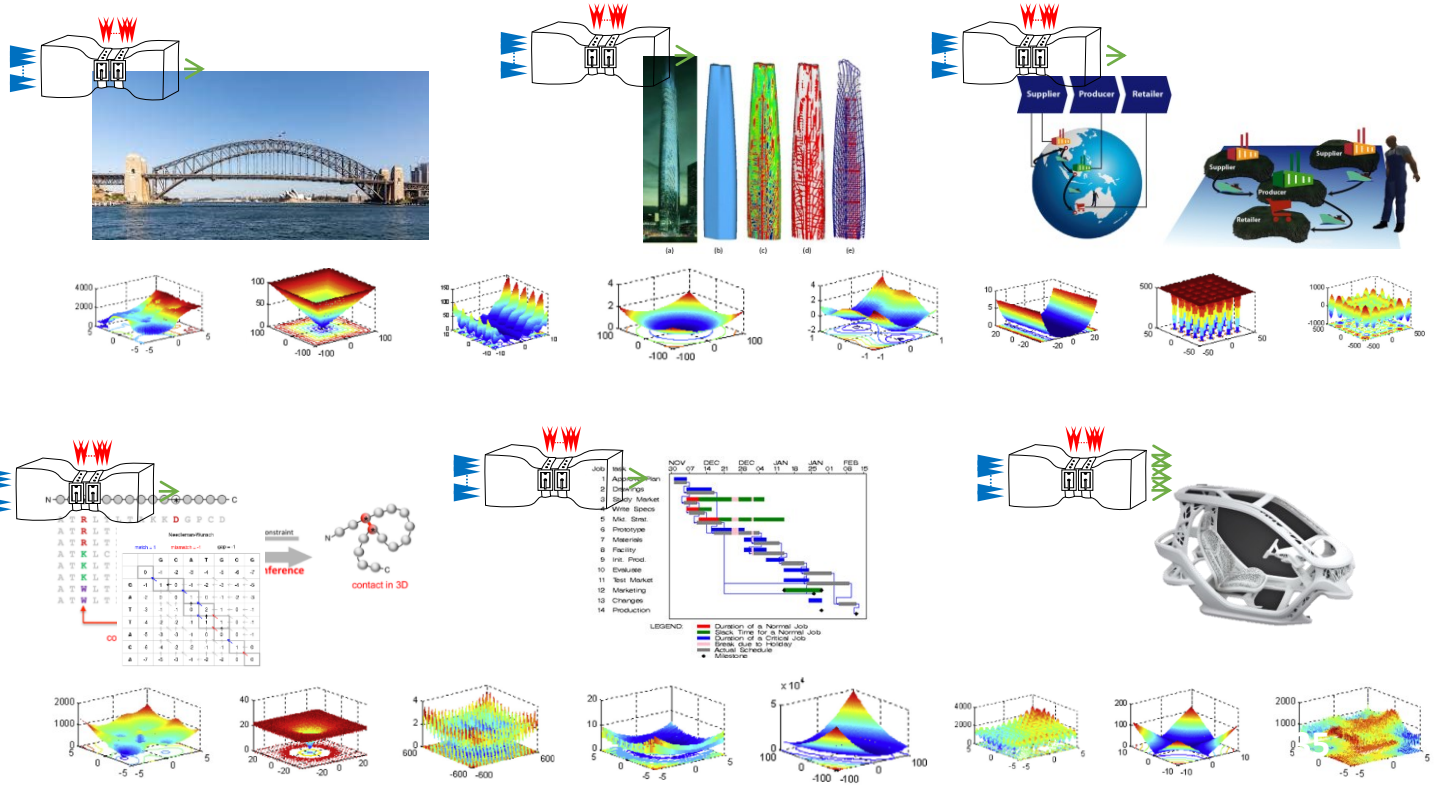
EXAMPLE 4: A NEW SET OF PERFORMANCE METRICS





THE RISK OF OVER OPTIMIZATION?

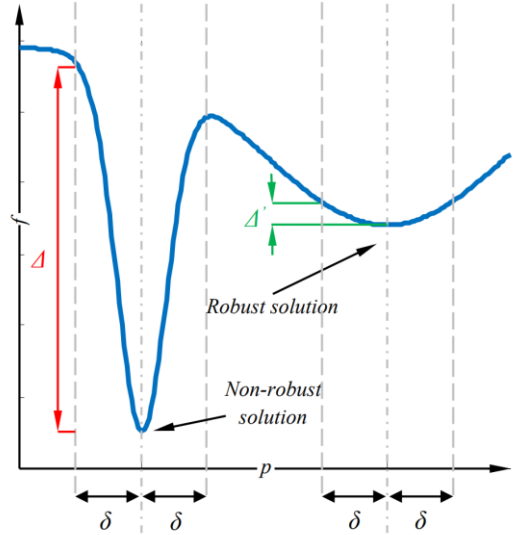
EVERYTHING CAN BE OPTIMIZED, BUT SHOULD WE?



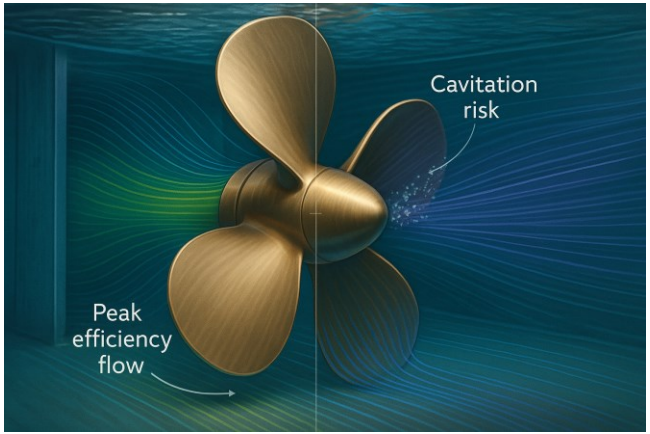
Human-generated images

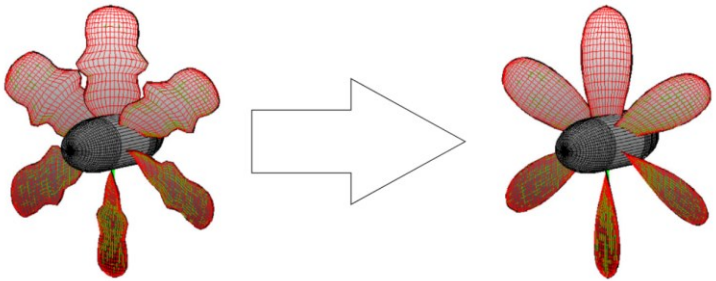
WHAT IF WE OVER OPTIMIZE?

- Small errors can lead to big failures: Tiny mistakes in how we set up the problem or prompt the model can grow into serious issues later, like wrong or unsafe solutions.
- Chain reaction of errors: If we use LLMs in several steps of optimization, a small error early on can spread and affect the final outcome without being noticed.
- Too tuned to the simulator: If we train or test the optimiser only using a simulator, it might learn tricks that work there but fail in the real world, because it's just fitting to the simulator's rounding errors or unrealistic assumptions.



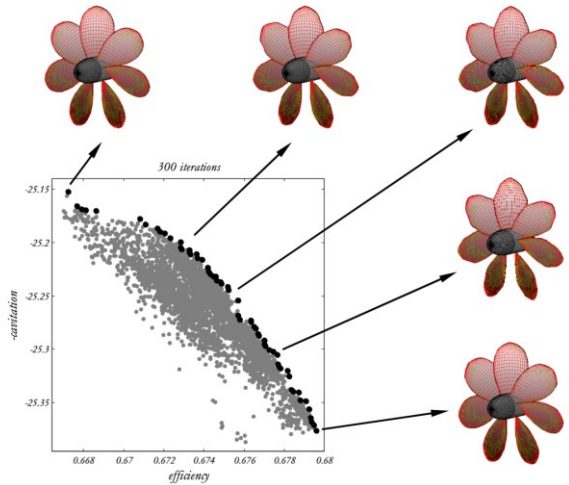
AN EXAMPLE

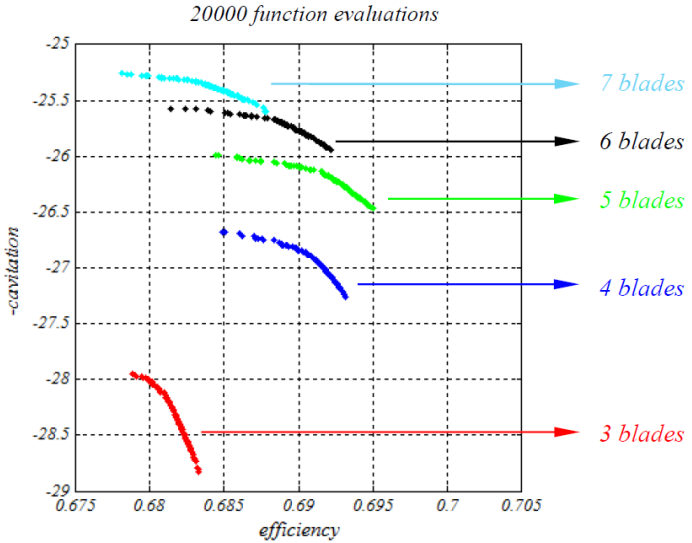


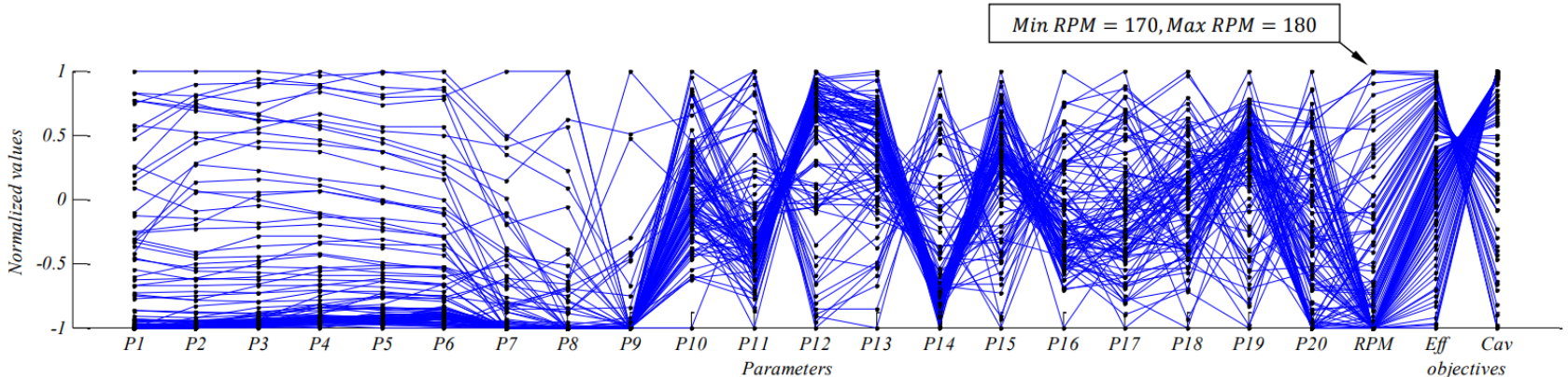


Propeller efficiency is the ratio of thrust power to shaft power; every extra percentage point is real fuel money(unless it comes with cavitation).

AN EXAMPLE


300 iterations
-cavitation
efficiency


20000 function evaluations
7 blades
6 blades
5 blades
4 blades
3 blades
-cavitation
efficiency


Min RPM = 170, Max RPM = 180
Normalized values
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20 RPM Eff Cav
Parameters
objectives

Effects of uncertainties in operating conditions on the objectives

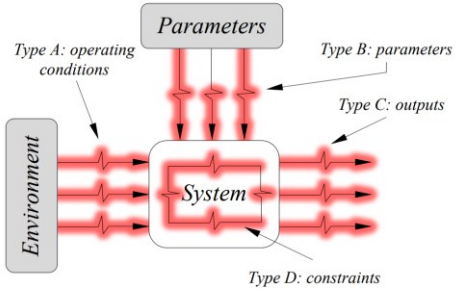


Figure 2.8: Different categories of uncertainties and their effects on a system: Type A, Type B, and Type C

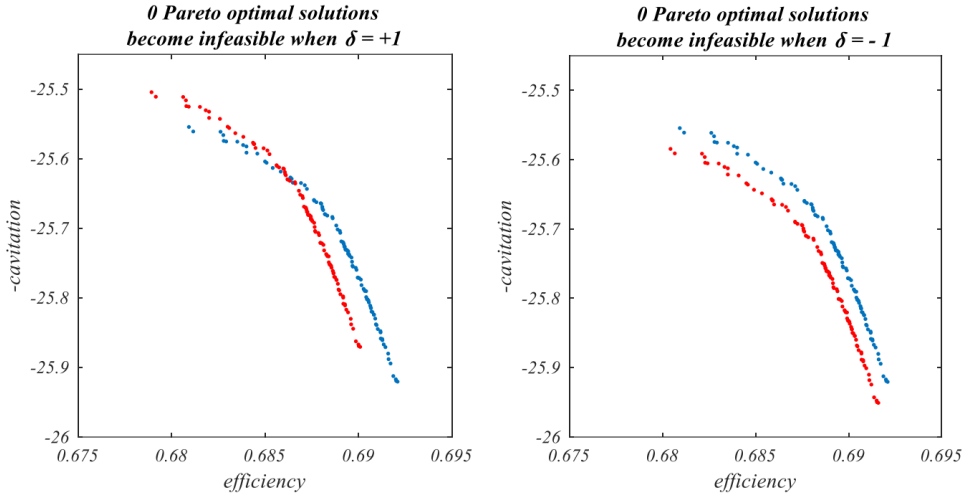


Figure 9.8: Pareto optimal solutions in case of (left) $\delta = +1.5\%$ (right) $\delta = -1.5\%$ perturbations in parameters. Original values are shown in blue, perturbed results in red.

Effects of uncertainties in operating conditions on the objectives

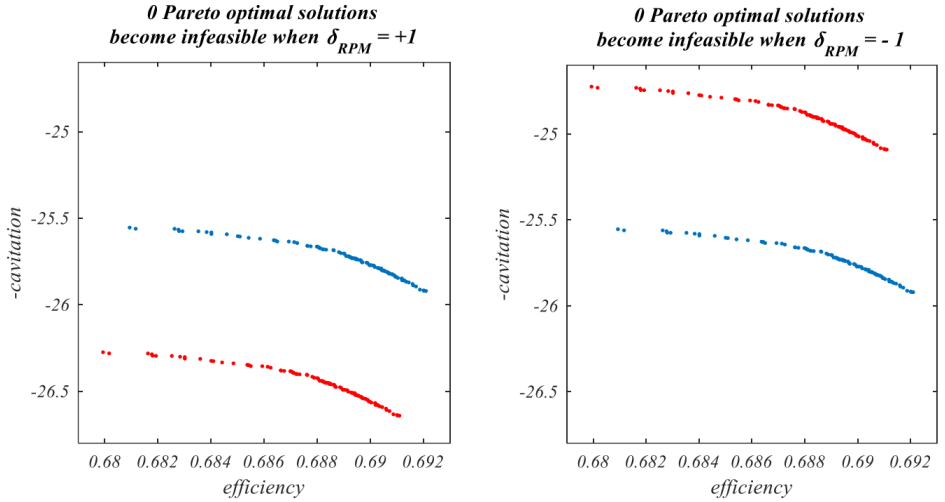


Figure 9.7: Pareto optimal solutions in case of (left) $\delta_{RPM} = +1$, (right) $\delta_{RPM} = -1$ fluctuations in RPM (right). Original values are shown in blue, perturbed results in red.

Perturbation may causse extra 40 liter of fuel consumption per day!!!

Considering 1.5% noise according to ISO 484/2-1981

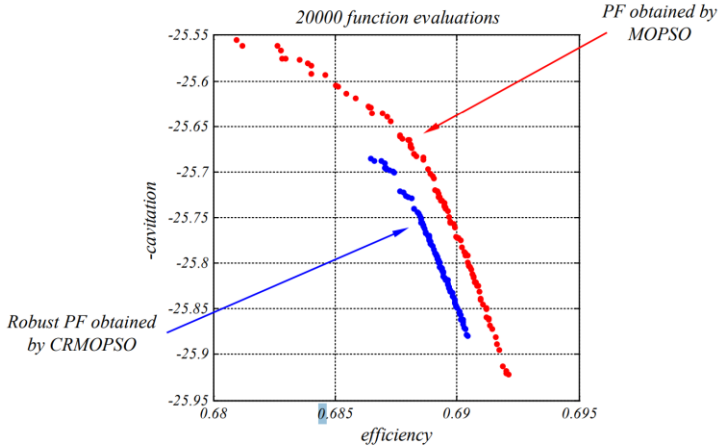


Figure 9.9: Robust front obtained by CRMOPSO versus global front obtained by MOPSO

Table 9.1: Fuel consumption discrepancy in case of perturbation in all of the structural parameters for both PS obtained by MOPSO and RPS obtained by CRMOPSO

Algorithm	average	min	max
MOPSO	0.1735	0.1676	0.1851
CRMOPSO	0.0825	0.0805	0.0863

CURRENT GAPS AND FUTURE DIRECTIONS

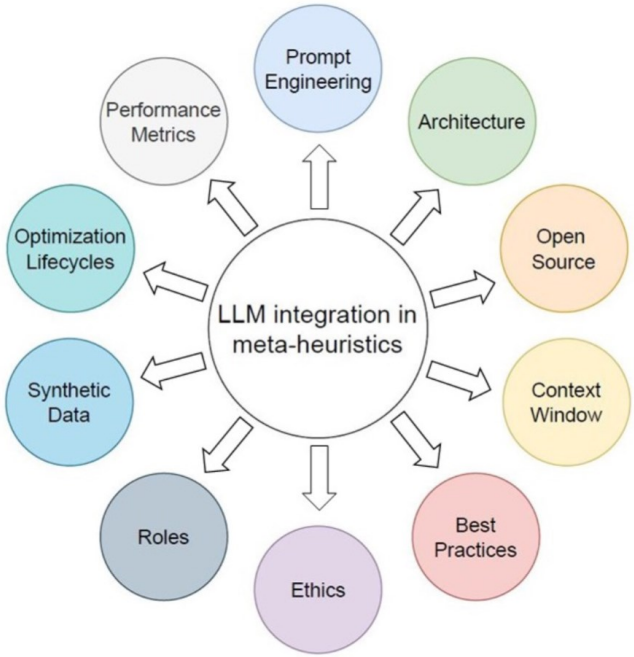
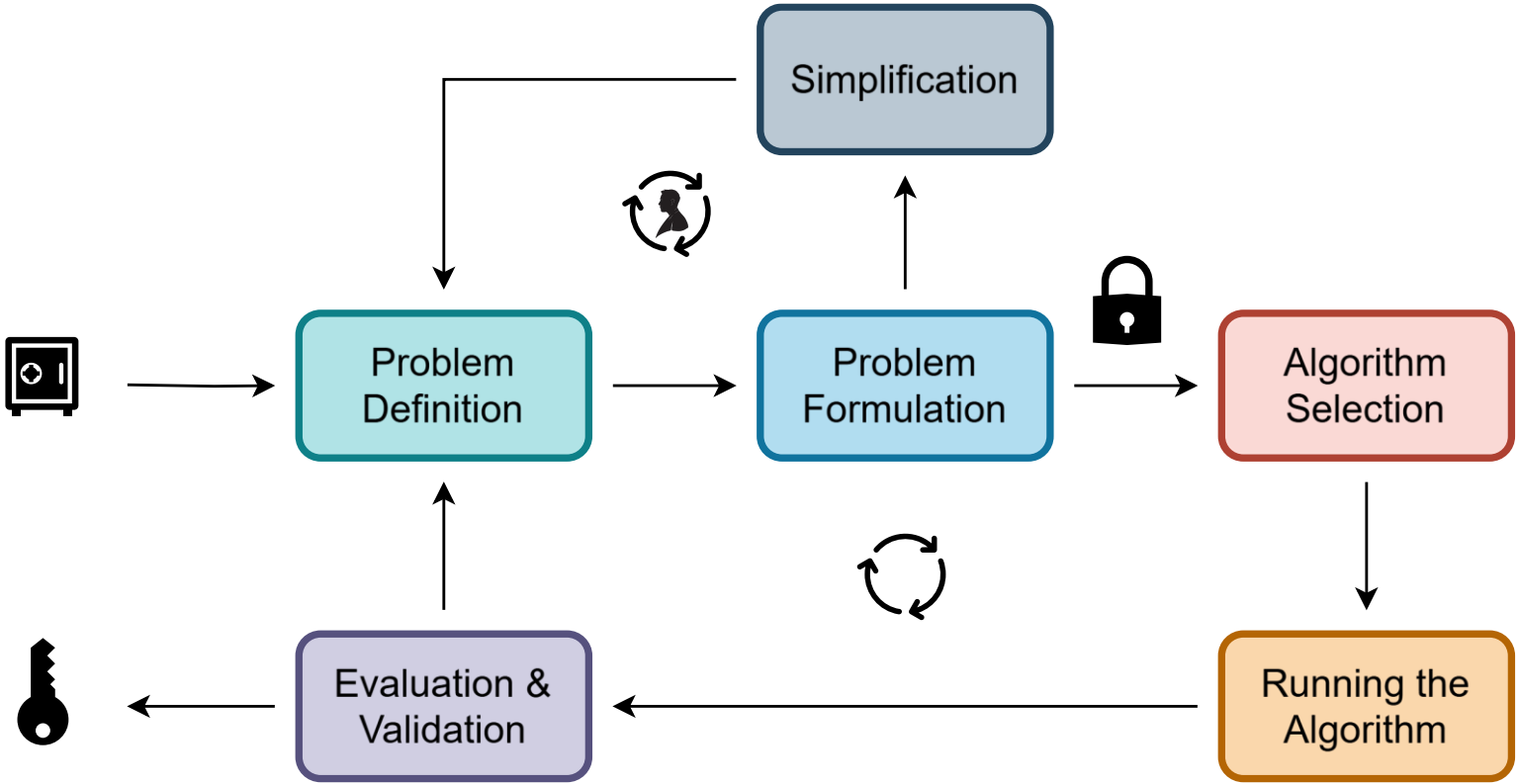


Figure 15: Suggested future research areas for using LLMs in metaheuristics. Each "dimension" represents a different focus area, such as creating specific frameworks, setting ethical guidelines, using synthetic data, and promoting open-source tools. The diagram highlights different paths researchers can take to advance research in this area.

WHAT TO DO TO MINIMIZE THE RISK?

OPTIMIZATION LIFECYCLE/WORKDFLOW MUSK STYLE

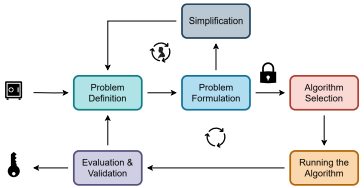


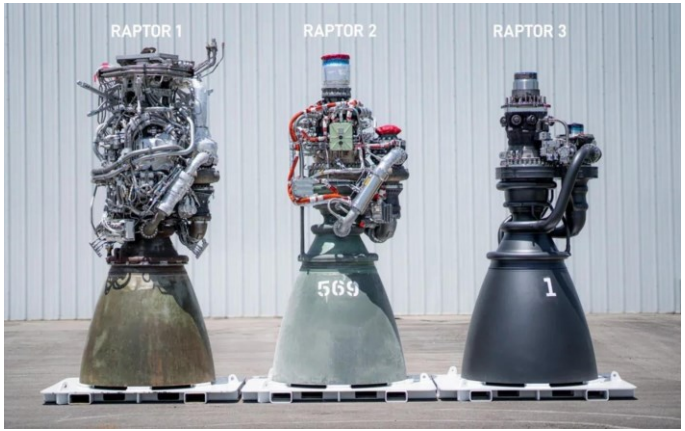
Human-generated figure

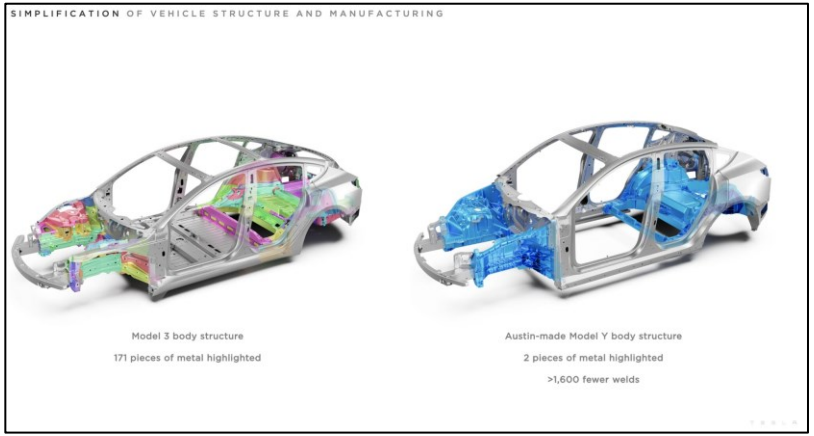
ELON MUSK's AGILE WORKFLOW

1. **Make the requirements less dumb**: "*your requirements are definitely dumb, it does not matter who gave them to you*", so question the question.
2. **Delete the part or process:** "*If you're not adding things back in at least 10% of the time, you're clearly not deleting enough*", so regularly review and remove parts or processes that don't add significant value.
3. **Simplify or optimize the design:** "*Possibly the most common error of a smart engineer is to optimize a thing that should not exist",* so focus on essential elements
4. **Accelerate cycle time:** *"Every process can be speeded up. But only do this after you have followed the first three steps. In the Tesla factory, I mistakenly spent a lot of time accelerating processes that I later realized should have been deleted."* , so focus on valuable progress instead of mere speed.
5. **Automate:** *"That comes last. The big mistake in Nevada and at Fremont was that I began by trying to automate every step. We should have waited until all the requirements had been questioned, parts and processes deleted, and the bugs were shaken out.",* so only automate after the first four steps.

EXAMPLES





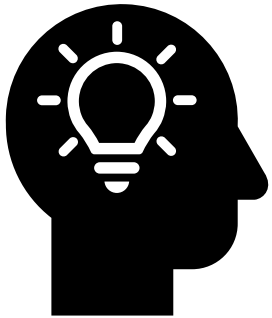


Source: Spacex.com and Tesla.com

TAKE AWAYS?

KEY TAKE AWAYS

1. GenAI can automate the whole optimization lifeycle
2. Early evidences are convincing (e.g. large-language-model guidance has already delivered >20X gains in PSO with zero human tweaking, hinting at the power of GenAI-designed optimizers)
3. Open challenges remain: fully automating algorithm improvement and problem formulation while keeping humans only where their insight truly matters.
4. Question requirements, delete what doesn’t add value, simplify first, speed up second, automate last.
5. Bottom line: anything can be optimized, but not everything should be. Beware the “too-perfect” solution. Over-optimization can make systems fragile and expensive when real-world noise or constraints shift.



VŠB TECHNICKÁ UNIVERZITA OSTRAVA

TORRENS UNIVERSITY AUSTRALIA

https://seyedalimirjalili.com

ali.mirjalili@gmail.com