KAMPUS BELA NEGARA

M1

FUNDAMENTALS OF HEURISTIC OPTIMIZATION

Concepts, principles, optimization problems, exact vs. heuristic approaches



Lecturer
Yisti Vita Via

Informatics
Faculty of Computer Science
UPN "Veteran" Jawa Timur

Course:
Heuristic Techniques

OUTLINE

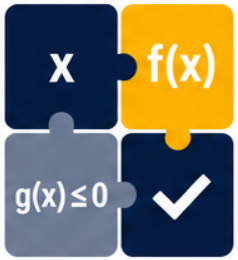
1 INTRODUCTION TO OPTIMIZATION

- Definition and purpose of optimization
- Optimization in Computer Science
- Real-world optimization problems



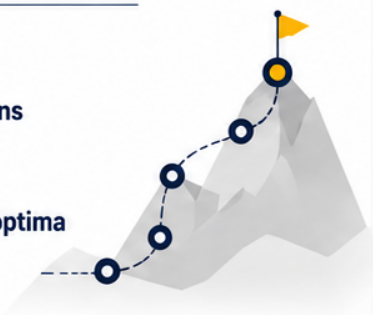
2 OPTIMIZATION PROBLEM COMPONENTS

- Decision variables
- Objective function
- Constraints
- Feasible and optimal solutions



3 SEARCH SPACE AND SOLUTIONS

- Search space
- Candidate solutions
- Solution quality
- Local and global optima



4 EXACT VS. HEURISTIC APPROACHES

- Exact optimization methods
- Heuristic approaches
- Strengths and limitations
- When to use heuristics



5 FUNDAMENTALS OF HEURISTIC OPTIMIZATION

- Heuristic principles
- Exploration and exploitation
- Search guidance
- Trade-off between solution quality and computational cost



6 ILLUSTRATIVE OPTIMIZATION PROBLEM

- Problem formulation
- Search process overview
- Interpreting heuristic solutions



1. INTRODUCTION TO OPTIMIZATION

Definition and purpose of optimization



Optimization is the process of finding the **best possible solution** from a set of feasible alternatives according to a given objective.

PURPOSE OF OPTIMIZATION

Maximize
achieve the highest possible value

Minimize
achieve the lowest possible value

Make Better Decisions
choose the most effective alternative

Use Resources Efficiently
reduce cost, time, and effort

EXAMPLES OF OPTIMIZATION

Logistics
Find the shortest route to deliver goods with minimum cost and time.

Manufacturing
Maximize production output while minimizing cost and material usage.

Finance
Maximize return on investment and minimize financial risk.

Human Resources
Optimize employee scheduling to improve productivity and satisfaction.

1. INTRODUCTION TO OPTIMIZATION

Optimization in Computer Science



Optimization is a fundamental concept in Computer Science used to **design efficient algorithms, manage resources, and solve complex problems.**

HOW OPTIMIZATION IS USED IN COMPUTER SCIENCE

Algorithm Design
Develop algorithms that are faster, smarter, and more efficient.

Resource Management
Efficiently allocate CPU, memory, storage, and network resources.

Combinatorial Optimization
Solve problems such as routing, scheduling, assignment, and clustering.

Machine Learning & AI
Optimize model parameters to improve accuracy and performance.

Security & Cryptography
Optimize encryption, authentication, and intrusion detection systems.

EXAMPLES

Shortest Path
Find the fastest route in a network.

Task Scheduling
Schedule tasks to minimize total completion time.

Load Balancing
Distribute workloads to maximize system performance.

Data Analysis
Select the best features or patterns from large datasets.

1. INTRODUCTION TO OPTIMIZATION

Real-world optimization problems



Optimization helps us make the best **decisions with limited resources** to **achieve the best possible outcomes** in real-world situations.

TYPES OF REAL-WORLD OPTIMIZATION PROBLEMS

Logistics & Transportation
Find the fastest or cheapest way to deliver goods to multiple locations.

Scheduling
Assign tasks, jobs, or resources to minimize time or cost.

Production Planning
Determine the best production plan to maximize profit and minimize waste.

Finance & Investment
Build an investment portfolio that maximizes return and minimizes risk.

Resource Allocation
Distribute limited resources (time, budget, staff) for maximum benefit.

EXAMPLES IN DAILY LIFE

Choosing the Best Route
Find the quickest route to avoid traffic and save time.

Budgeting
Allocate money across needs and wants to get maximum satisfaction.

Seat Selection
Choose the best available seat for comfort and convenience.

Energy Management
Use energy efficiently to maximize device battery life.

Team Formation
Form the best team with the right skills to achieve goals.

2. OPTIMIZATION PROBLEM COMPONENTS

Decision variables



Decision variables are the **unknowns** or **choices** that we can **control** in a problem.

We adjust their values to achieve the **best outcome.**

STEPS TO IDENTIFY DECISION VARIABLES

1. **Understand the problem**
   Identify the goal, given information, and key constraints.
2. **Determine the choices**
   Find the quantities or actions that we can control.
3. **Define the variables**
   $x_1, x_2, \ldots, x_n$
   Assign symbols to represent each decision variable.
4. **Set the range (if known)**
   Determine the possible values or limits of each variable.

EXAMPLES OF DECISION VARIABLES

1. Delivery Routing



Problem: Find the best routes for several delivery vehicles.

Decision variables:
$x_{ij} = 1$ if vehicle i goes from location i to j, 0 otherwise

Meaning: Decide which route each vehicle will take.

2. Production Planning



Problem: Decide how many units of each product to produce.

Decision variables:
x_i = number of units of product i

Meaning: Decide how much of each product will be produced.

3. Task Scheduling



Problem: Schedule tasks on machines to minimize total completion time.

Decision variables:
s_i = start time of task i

Meaning: Decide when each task will start.

4. Investment Planning



Problem: Decide how much money to invest in different assets.

Decision variables:
x_i = amount invested in asset i

Meaning: Decide how much to invest in each asset.

2. OPTIMIZATION PROBLEM COMPONENTS

Objective function



The objective function is a **mathematical expression** that measures the goal we want to achieve.

We **optimize** (**maximize** or **minimize**) this function while satisfying the given constraints.

HOW TO DEFINE AN OBJECTIVE FUNCTION

1 Understand the goal



Know what you want to achieve.

2 Choose what to measure



Select a metric that represents the goal.

3 Build the mathematical expression

Σ

Formulate the function using decision variables.

4 Determine direction

Maximize (Max) | Minimize (Min)

Decide whether to maximize or minimize the function.

EXAMPLES OF OBJECTIVE FUNCTIONS

1 Profit Maximization



Maximize total profit.

$$\text{Max } Z = \sum_{i=1}^{n} p_i x_i - \sum_{i=1}^{n} c_i x_i$$

Where:

- p_i = price of product i
- c_i = cost of producing product i
- x_i = amount of product i

Goal: Get the highest profit.

2 Cost Minimization



Minimize total cost.

$$\text{Min } Z = \sum_{i=1}^{n} c_i x_i$$

Where:

- c_i = cost per unit of resource i
- x_i = amount of resource i used

Goal: Spend the least cost.

3 Time Minimization



Minimize total time.

$$\text{Min } Z = \sum_{i=1}^{n} t_i x_i$$

Where:

- t_i = time required for activity i
- x_i = amount of activity i

Goal: Finish in the shortest time.

4 Quality Maximization



Maximize quality or performance.

$$\text{Max } Z = \sum_{i=1}^{n} q_i x_i$$

Where:

- q_i = quality score of option i
- x_i = level or amount of option i

Goal: Achieve the best quality.

2. OPTIMIZATION PROBLEM COMPONENTS

Constraints



Constraints are the limitations or rules that restrict the possible values of decision variables. They represent real-world limits such as resources, capacity, budget, or time.

TYPES OF CONSTRAINTS -

Less Than or Equal (≤)

The total used cannot exceed the available limit.

Example:
$2x + 3y \leq 100$
(total resource cannot exceed 100)

Greater Than or Equal (≥)

The total must be at least the required amount.

Example:
$x + y \geq 50$
(minimum demand must be met)

Equal (=)

The total must be exactly equal to a specific value.

Example:
$x + y = 80$
(must use exactly 80 units)

EXAMPLES OF CONSTRAINTS IN REAL-WORLD PROBLEMS -

1. Production Planning



Constraint:
Total available machine time is limited.

Example:
$3x_1 + 2x_2 \leq 120$
(total machine hours cannot exceed 120)

2. Delivery Routing



Constraint:
Each vehicle has a maximum capacity.

Example:
$x_1 + x_2 + x_3 \leq 100$
(total load cannot exceed 100 kg)

3. Budget Allocation



Constraint:
Total budget cannot exceed the available budget.

Example:
$5x_1 + 3x_2 + 4x_3 \leq 1000$
(total cost cannot exceed $1000)

4. Task Scheduling



Constraint:
Each machine can work only a limited number of hours per day.

Example:
$x_1 + x_2 + x_3 \leq 8$
(total working hours cannot exceed 8 hours)

2. OPTIMIZATION PROBLEM COMPONENTS

Feasible and optimal solutions

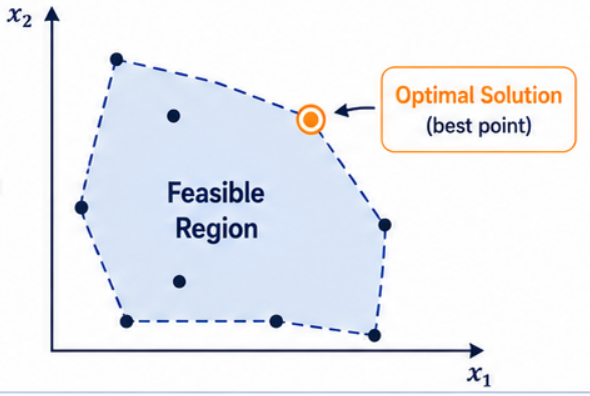


A **feasible solution** is any solution that satisfies all the constraints.

An **optimal solution** is a feasible solution that gives the **best value** (may be maximum or minimum) of the objective function.

VISUAL EXPLANATION

- **Feasible Region** (Satisfies all constraints)
- **Feasible Solution** (Any point inside the region)
- **Optimal Solution** (Best feasible solution according to the objective function)



KEY CONCEPTS

Feasible Solution

- Satisfies all the problem constraints.
- There can be many feasible solutions.
- Not all feasible solutions are the best.

Optimal Solution

- Is a feasible solution.
- Gives the best (maximum or minimum) value of the objective function.
- There can be one or more optimal solutions.

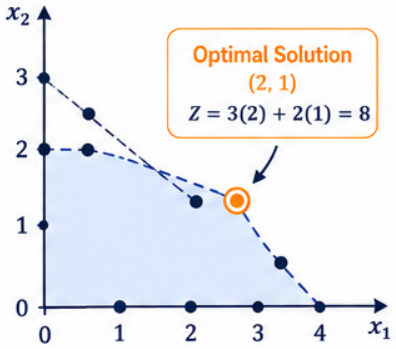
EXAMPLE

Maximize $Z = 3x_1 + 2x_2$

Subject to:

$$x_1 + x_2 \leq 4$$
$$2x_1 + x_2 \leq 5$$
$$x_1 \geq 0, x_2 \geq 0$$

- Feasible Region
- Other Feasible Solutions
- Optimal Solution



SUMMARY

Feasible solutions

Meet all constraints. They are inside the feasible region.

Optimal solutions

Are the best feasible solutions according to the objective function.

3. SEARCH SPACE AND SOLUTIONS

Search space



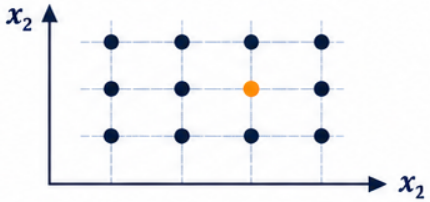
The search space is the set of all possible solutions (combinations of decision variables) that we can consider in an optimization problem.

Finding the best solution means searching in this space.

TYPES OF SEARCH SPACE

Discrete Search Space

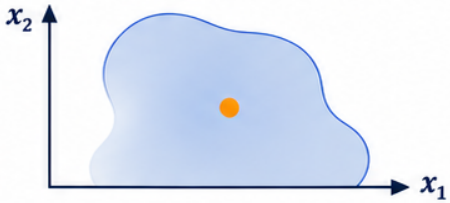
Solutions are countable (finite or infinite but separate).



Example: Number of items to produce, routing choices, scheduling order.

Continuous Search Space

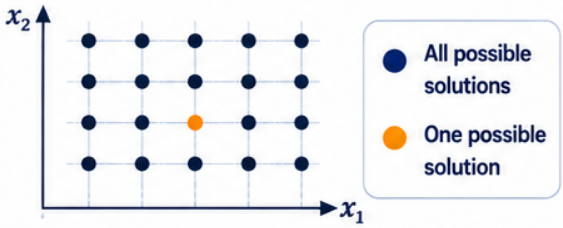
Solutions are real numbers within a range.



Example: Amount of resources, time allocation, investment proportion.

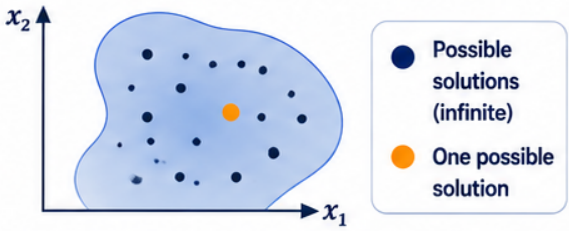
VISUALIZING THE SEARCH SPACE

1. Discrete Search Space (2 Variables)



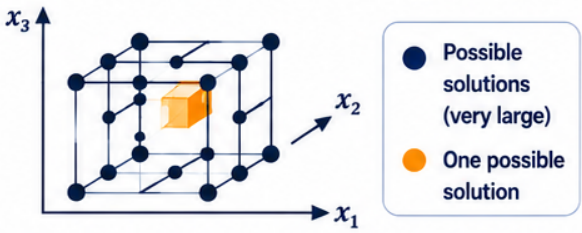
Each point represents a specific solution (x_1, x_2) from the set of discrete choices.

2. Continuous Search Space (2 Variables)



The region contains infinitely many possible solutions within given variable ranges.

3. High-Dimensional Search Space (n Variables)



As the number of variables increases, the search space grows exponentially.

3. SEARCH SPACE AND SOLUTIONS

Candidate solutions



Candidate solutions are **potential solutions** that we generate or consider while searching the search space.

Not all candidate solutions are feasible, and even fewer are optimal.

ROLE IN THE SEARCH PROCESS

1 Generate



Create a set of candidate solutions from the search space.

2 Evaluate



Check feasibility and evaluate the objective function.

3 Compare



Compare candidate solutions to identify the best so far.

4 Select / Update



Select the best candidate or update the search direction.

EXAMPLE

Problem: Choose how many units of Product A and Product B to produce to maximize profit.

Decision variables:

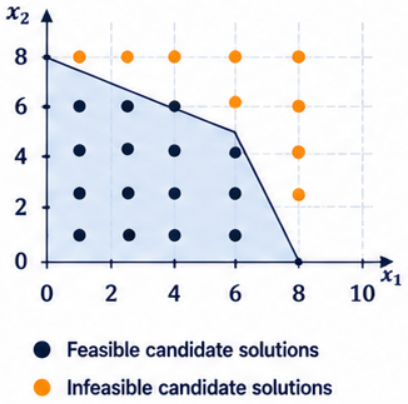
- x_1 = units of Product A
- x_2 = units of Product B

Constraints:

- $2x_1 + x_2 \leq 10$ (Resource 1)
- $x_1 + 2x_2 \leq 8$ (Resource 2)
- $x_1, x_2 \geq 0$

Objective: Maximize $Z = 5x_1 + 4x_2$

Search Space (x_1, x_2)



List of candidate solutions (examples)

Candidate	(x_1, x_2)	Feasible?	$Z = 5x_1 + 4x_2$
A	(0, 0)	✓	0
B	(2, 1)	✓	14
C	(3, 2)	✓	23
D	(4, 0)	✓	20
E	(6, 2)	✗	38
F	(5, 4)	✗	41
G	(1, 5)	✗	25

Best candidate so far

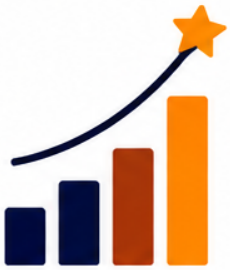
Candidate C (3, 2)

$Z = 23$

The search algorithm will continue exploring other candidate solutions to find the optimal one.

3. SEARCH SPACE AND SOLUTIONS

Solution quality



Solution quality refers to how good a candidate solution is with respect to the objective function. We aim to find the **best (optimal) solution**, but in practice we may settle for the best solution found so far.

HOW WE MEASURE QUALITY

Objective Value

The value of the objective function for a solution. (For maximization: higher is better. For minimization: lower is better.)

Feasibility

A solution is feasible if it satisfies all constraints. Infeasible solutions, no matter how good their objective value, are invalid.

Optimality / Suboptimality

Optimal solution is the best possible solution. Suboptimal solution is close to optimal but not the best.

EXAMPLE

Problem: Choose how many units of Product A and Product B to produce to maximize profit.

Decision variables:

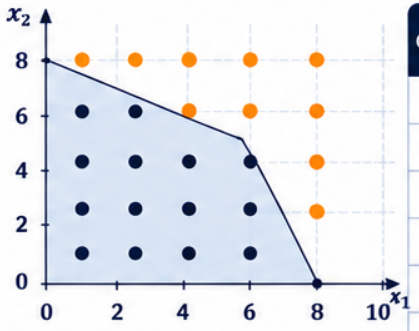
- x_1 = units of Product A
- x_2 = units of Product B

Constraints:

- $2x_1 + x_2 \leq 10$ (Resource 1)
- $x_1 + 2x_2 \leq 8$ (Resource 2)
- $x_1, x_2 \geq 0$

Objective: Maximize $Z = 5x_1 + 4x_2$

Search Space (x_1, x_2)



- Feasible candidate solutions
- Infeasible candidate solutions

Evaluate solution quality

Candidate	(x_1, x_2)	Feasible?	$Z = 5x_1 + 4x_2$ (Objective value)	Quality (Maximization)
A	(0, 0)	✓	0	Poor
B	(2, 1)	✓	14	Fair
C	(3, 2)	✓	23	Good
D	(4, 0)	✓	20	Fair
E	(6, 2)	✗	38	– (Infeasible)
F	(5, 4)	✗	41	– (Infeasible)
G	(1, 5)	✗	25	– (Infeasible)

Best solution so far

Candidate C (3, 2)

$Z = 23$

(Feasible and highest objective value)

The search algorithm aims to improve solution quality by exploring the search space.

3. SEARCH SPACE AND SOLUTIONS

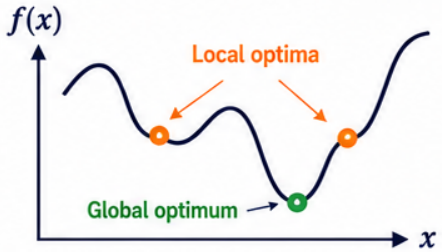
Local and global optima



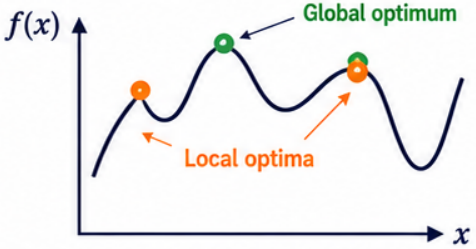
In an optimization landscape, a **local optimum** is the best solution in a neighborhood, while a **global optimum** is the best solution in the entire search space.

UNDERSTANDING THE CONCEPT

Minimization Problem



Maximization Problem



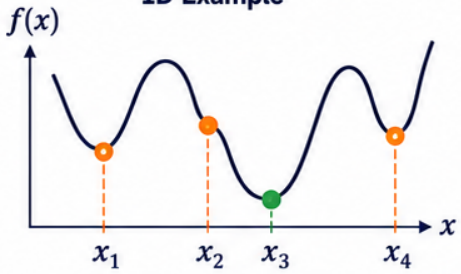
Many optimization algorithms may get stuck at a local optimum.
The goal is to find the **global optimum**.

EXAMPLE

Objective: Minimize $f(x)$
(Lower value is better)

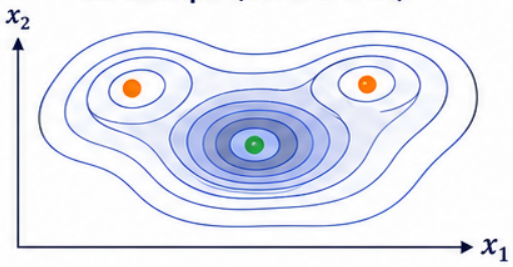
- Local optimum
- Global optimum

1D Example



x_1, x_3, x_4 are **local optima**
x_2 is the **global optimum**

2D Example (Contour Plot)



Orange points: local optima
Green point: global optimum

KEY TAKEAWAY

Local optimum is good in its neighborhood.

Global optimum is the best overall.

Optimization aims to find the **global optimum**.

4. EXACT VS. HEURISTIC APPROACHES

Exact optimization methods

Exact optimization methods are algorithms that **always find the optimal solution** (if one exists) by systematically exploring the search space.

They guarantee optimality, but often require high computational effort.

HOW EXACT METHODS WORK

1. **Model the problem** — Formulate objective function and constraints.
2. **Explore the search space** — Systematically explore all feasible solutions or use mathematical techniques.
3. **Evaluate solutions** — Compute objective value for each feasible solution.
4. **Identify the best solution** — Select the solution with the best objective value.
5. **Prove optimality** — Use mathematical proof (e.g., optimality gap = 0) to confirm optimality.

EXAMPLES

Common exact optimization methods:

- Linear Programming (Simplex, Interior-Point)
- Integer Linear Programming (Branch and Bound)
- Dynamic Programming
- Exhaustive Enumeration (for small problems)



Example:
In a scheduling problem with 10 tasks, Branch and Bound systematically explores the search tree and guarantees the best possible schedule.

ADVANTAGES

- Guarantees the optimal solution (if a feasible solution exists).
- Provides a bound on solution quality during the search.
- Reliable and exact.

DISADVANTAGES

- Computationally expensive for large problems.
- May require huge memory and long running time.
- Not scalable to very large search spaces.

4. EXACT VS. HEURISTIC APPROACHES

Heuristic approaches

Heuristic approaches use rules of thumb, experience, or problem-specific insights to find good (not necessarily optimal) solutions quickly, especially for large or complex problems.

They focus on finding good solutions in reasonable time rather than guaranteeing the best solution.

HOW HEURISTIC APPROACHES WORK

1. **Understand the problem** — Identify key factors and constraints.
2. **Design a heuristic** — Create rules or strategies to build solutions.
3. **Generate solutions** — Construct one or more candidate solutions.
4. **Evaluate solutions** — Assess the quality of solutions using the objective function.
5. **Improve iteratively** — Refine or modify solutions to find better ones.

EXAMPLES OF HEURISTIC APPROACHES

Greedy Heuristic

Make the locally best choice at each step with the hope of finding a good overall solution.

Constructive Heuristic

Build a solution step by step using simple rules.

Local Search

Start with an initial solution and iteratively move to neighboring solutions to improve quality.

Metaheuristics

High-level strategies that guide and combine other heuristics (e.g., Genetic Algorithm, Simulated Annealing).

ADVANTAGES

- Fast and scalable for large problems.
- Flexible and easy to adapt.
- Can find very good solutions in practice.

DISADVANTAGES

- No guarantee of optimality.
- Solution quality depends on the heuristic design.
- May get stuck in local optima.

EXAMPLE: In the Traveling Salesman Problem with thousands of cities, heuristic methods (e.g., nearest neighbor, 2-opt local search) can quickly find a very good tour, while exact methods may take too long or become impossible.

Heuristic approaches trade optimality for speed and scalability, aiming to find high-quality solutions within practical time limits.

4. EXACT VS. HEURISTIC APPROACHES

Strengths and limitations

Both exact methods and heuristic approaches have their own strengths and limitations. The choice depends on the problem characteristics and practical requirements.

No single approach is best for every problem. Choose the right tool for the right problem.

In practice, hybrid approaches often combine the strengths of both worlds.

ASPECT	EXACT OPTIMIZATION METHODS	HEURISTIC APPROACHES
Solution Quality	✓ Guarantees the optimal solution (if a feasible solution exists). ✗ None – always optimal (but may be hard to find).	✓ Often finds very good (near-optimal) solutions. ✗ No guarantee of optimality.
Computational Effort	✓ Systematic and thorough. ✗ Can be computationally expensive, especially for large problems.	✓ Generally faster and more efficient. ✗ May require many iterations to improve.
Scalability	✓ Works well for small to medium problems. ✗ Does not scale well to very large search spaces.	✓ Scales well to large and complex problems. ✗ Performance may degrade on extremely large problems.
Reliability	✓ Highly reliable and predictable. ✗ Sensitive to modeling errors and numerical issues.	✓ Less predictable; performance depends on design. ✗ Solution quality can vary between runs.
Implementation	✓ Well-established algorithms and software. ✗ May be complex to implement and tune.	✓ Often simpler to implement. ✗ Requires careful design of heuristics and parameters.
When to Use	✓ When optimality is critical and problem size is small or medium.	✓ When quick, good-enough solutions are needed, especially for large or complex problems.

Exact methods: Quality guaranteed, cost may be high.

Heuristics: Good results fast, but no guarantee.

4. EXACT VS. HEURISTIC APPROACHES

When to use heuristics



Heuristics are not always the best choice, but they are often the most practical option.

Use heuristics when the problem is too large, too complex, or time-critical for exact methods.

KEY IDEA



Use heuristics when you need good solutions quickly and can accept near-optimal results.

Speed and scalability are priorities.

USE HEURISTICS WHEN...

1. **The problem size is very large**



Exact methods may take too much time or memory to solve large problems.

2. **The search space is too large to explore**



It is impossible or impractical to evaluate all possible solutions.

3. **Time is limited**



A good solution found quickly is better than the optimal solution found too late.

4. **Solutions can be near-optimal**



A near-optimal solution is good enough for practical purposes.

5. **The problem is complex or poorly understood**



Heuristics can work well even when the problem structure is unclear.

6. **The problem changes frequently**



Heuristics can adapt quickly to dynamic environments.

EXAMPLE SCENARIOS

- **Routing and Logistics:** Find good delivery routes among thousands of locations.
- **Scheduling:** Assign jobs or resources in factories, hospitals, or airlines.
- **Design Problems:** Optimize complex engineering or architectural designs.
- **AI and Machine Learning:** Train models or search for good features/parameters.
- **Games and Planning:** Make good moves in chess, Go, or real-time strategy games.



RULE OF THUMB

If exact methods are too slow or impossible, and a good (not perfect) solution is acceptable, then use heuristics.

5. FUNDAMENTALS OF HEURISTIC OPTIMIZATION

Heuristic principles

Heuristics are guided strategies that use problem-specific knowledge, experience, and simple rules to explore the search space efficiently and find good (near-optimal) solutions.

Heuristics do not guarantee optimality, but they aim for good solutions in reasonable time.

CORE PRINCIPLES OF HEURISTIC OPTIMIZATION

1. **Intensification** — Focus on promising areas of the search space to find better solutions.
2. **Diversification** — Explore different areas to avoid getting trapped in local optima.
3. **Guided Search** — Use information (heuristics) to guide the search toward better solutions.
4. **Trade-off** — Balance exploration (diversification) and exploitation (intensification).
5. **Iterative Improvement** — Improve solutions step by step until stopping criteria are met.

HOW HEURISTICS WORK

1. **Start with an initial solution** — Generate a feasible solution to begin the search.
2. **Evaluate the solution** — Measure the quality of the solution using an objective function.
3. **Generate candidate solutions** — Create new solutions by applying heuristic rules or operators.
4. **Select the best candidate(s)** — Choose the most promising solution(s) to continue.
5. **Update and repeat** — Replace the current solution and repeat until stopping condition is met.

Heuristics continuously search for better solutions using feedback from previous evaluations.

KEY CHARACTERISTICS

Problem-specific	Designed using knowledge about the problem structure and constraints.
Efficient	Require less time and memory compared to exact methods.
Approximate	Aim for near-optimal solutions, not guaranteed optimal solutions.
Flexible	Can be adapted to different problems and constraints.
Scalable	Capable of handling large and complex search spaces.

THE GOAL: Apply simple rules and smart strategies to search effectively, achieve good solutions quickly, and solve complex problems in practice.

REMEMBER: Heuristics are about finding good solutions, not always the best one.

5. FUNDAMENTALS OF HEURISTIC OPTIMIZATION

Exploration and exploitation



Heuristic optimization is all about balancing exploration and exploitation. Exploration discovers new promising areas, while exploitation makes the most of known good areas.

A good balance helps find high-quality solutions without getting stuck in local optima.

THE BALANCE

EXPLORATION
(Search widely)

The goal is to find the best solutions, not just any solution.

EXPLOITATION
(Search deeply)



Look for new regions and diverse solutions. Discover opportunities.

Focus on the best known solutions. Refine and improve.

EXPLORATION VS. EXPLOITATION SPECTRUM

More Exploration ← Balanced → More Exploitation

- Search broadly, try many different possibilities. Higher diversity, higher risk.
- Explore enough to understand the landscape before focusing.
- Focus on the best areas found so far. Improve current solutions.

THE TRADE-OFF

Too Much Exploration	VS.	Too Much Exploitation
Wastes time and computational resources.		May get stuck in local optima.
May never focus on good solutions.		Misses better solutions in other regions.
Slow convergence.		Poor overall performance.

EXAMPLES



Exploration (Hiking Analogy):
You hike to new, unexplored trails to discover new views and hidden places.



Exploitation (Hiking Analogy):
You stay on the best trail you've found and go deeper to reach the summit faster.

KEY TAKEAWAYS

- Start with more exploration, then gradually shift toward exploitation.
- Use adaptive strategies to balance exploration and exploitation.
- The right balance depends on the problem and the stage of the search.



5. FUNDAMENTALS OF HEURISTIC OPTIMIZATION

Search guidance



Search guidance refers to the information, strategies, or rules used to direct the search process toward better (promising) solutions and away from unpromising areas.

Good guidance = faster search, better solutions, and less computation.

WHY SEARCH GUIDANCE MATTERS

Focus the Search
Narrows the large search space to the most promising regions.

Improve Efficiency
Reduces time and computational effort by avoiding useless exploration.

Enhance Solution Quality
Guides the search toward high-quality (near-optimal) solutions.

Increase Robustness
Helps the search adapt to different problem types and conditions.

COMMON SOURCES OF SEARCH GUIDANCE

1. **Heuristics / Rules of Thumb**
   Simple, problem-specific rules or intuitive knowledge that estimate what is good.
2. **Evaluation / Fitness Function**
   Scores candidate solutions or states to estimate how good they are.
3. **Problem Knowledge**
   Understanding of the problem structure, constraints, and relationships.
4. **Historical Information**
   Information from past searches or solutions to avoid repeating mistakes.

GOOD SEARCH GUIDANCE IS...

- **Informative:** Provides useful information about solution quality.
- **Directional:** Points the search toward better areas.
- **Efficient:** Easy to compute and update.
- **Balanced:** Helps explore enough while focusing on good solutions.
- **Adaptive:** Adjusts as the search progresses.

EXAMPLES OF SEARCH GUIDANCE IN PRACTICE

Routing Problems
Use distance, traffic estimates, or shortest path heuristics.

Scheduling
Use earliest finish time, least slack, or priority rules.

Design Problems
Use constraints, evaluation functions, or simulation results.

Machine Learning Optimization
Use loss function, gradients, or surrogate models.

AI / Game Search
Use evaluation functions, heuristic estimates, or pattern knowledge.

The better the search guidance, the faster and smarter the search.

Good guidance does not guarantee the best solution, but it greatly increases the chance of finding a good one.

5. FUNDAMENTALS OF HEURISTIC OPTIMIZATION

Trade-off between solution quality and computational cost



In heuristic optimization, improving solution quality usually requires more computational time and resources.

The key is to find the right balance that fits the problem and the available resources.

There is no one-size-fits-all. The best choice depends on the goals, constraints, and context of the problem.

THE TRADE-OFF CONTINUUM

Higher Computational Cost (More Time & Resources)

Choose the right balance based on your goals and constraints.

Lower Computational Cost (Less Time & Resources)

- More iterations / evaluations
- More complex strategies
- More accurate / higher quality solutions

- Fewer iterations / evaluations
- Simpler strategies
- Faster, but lower quality solutions

IMPACT OF THE TRADE-OFF

Focus on Higher Quality		VS.	Focus on Lower Cost	
Computational Cost	High		Computational Cost	Low
Solution Quality	High		Solution Quality	Lower
Risk	Low (better solutions)		Risk	High (may miss better solutions)
Use When	Accuracy is critical, resources are available, and time is less constrained.		Use When	Quick decisions are needed, resources are limited, or approximate solutions are acceptable.

EXAMPLES OF THE TRADE-OFF IN PRACTICE

		VS.	
Traveling Salesman Problem (TSP)	Exact methods (e.g., Branch and Bound) — High quality, High cost	VS.	Heuristics (e.g., Genetic Algorithm) — Good quality, Low cost
Job Shop Scheduling	Complex metaheuristics (e.g., Simulated Annealing) — Higher quality, Higher cost	VS.	Simple heuristics (e.g., FIFO, SPT) — Lower quality, Lower cost
Machine Learning Hyperparameter Tuning	Extensive search (e.g., Bayesian Optimization) — Better performance, Higher cost	VS.	Random / Simple search (e.g., Random Search) — Good enough, Lower cost

TAKEAWAY: Better solutions often cost more time and resources. Pick the balance that delivers the most value for your specific situation.

REMEMBER: The goal is not always the best possible solution, but the best solution within your constraints.

6. ILLUSTRATIVE OPTIMIZATION PROBLEM

Problem formulation



Problem formulation is the first and most important step in solving an optimization problem. It **defines what we want to optimize, the decisions we can make,** and the limitations we must respect.

EXAMPLE PROBLEM



Store Location Problem

A company wants to open new stores in a city. The goal is to choose the best locations to maximize expected profit while considering budget and other constraints.

1. DECISION VARIABLES

The choices or decisions that can be made.



Let x_i =
- 1 if a store is opened at location i
- 0 otherwise

$i = 1, 2, \ldots, n$
(n = number of candidate locations)

2. OBJECTIVE FUNCTION

The goal is to maximize (or minimize) a quantity of interest.



Maximize total expected profit:

$$\text{Maximize } Z = \sum_{i=1}^{n} p_i x_i$$

where p_i = expected profit from opening a store at location i

3. CONSTRAINTS

The limitations or restrictions that must be satisfied.

Budget constraint	$\sum_{i=1}^{n} c_i x_i \le B$	Total cost must not exceed budget B
Market coverage constraint	$\sum_{i=1}^{n} a_{ji} x_i \ge D_j$	Minimum demand D_j must be covered in area j
Limit on number of stores	$\sum_{i=1}^{n} x_i \le K$	At most K stores can be opened
	$x_i \in \{0, 1\}$ (Decision variables are binary)	A store is either opened (1) or not (0)

4. COMPLETE MATHEMATICAL MODEL

Objective: $\text{Maximize } Z = \sum_{i=1}^{n} p_i x_i$

Subject to:

$\sum_{i=1}^{n} c_i x_i \le B$ (Budget)

$\sum_{i=1}^{n} a_{ji} x_i \ge D_j$ (Market coverage, $\forall j$)

$\sum_{i=1}^{n} x_i \le K$ (Max number of stores)

Decision variables: $x_i \in \{0, 1\}, \forall i = 1, 2, \ldots, n$ (Binary decisions)



This model clearly states:
- What we want to achieve (objective)
- What choices we can make (decision variables)
- What rules we must follow (constraints)

KEY TAKEAWAYS



- A good formulation makes the problem clear and solvable.
- It is the foundation for applying exact or heuristic methods.
- Time spent on good formulation saves time in solving!

6. ILLUSTRATIVE OPTIMIZATION PROBLEM

Search process overview



The search process is the systematic exploration of the solution space to find a high-quality (or optimal) solution that satisfies all constraints.

Heuristics guide the search toward promising solutions efficiently.

SEARCH AT A GLANCE

Start — Initial Solution

→ **Explore** — Generate and evaluate neighboring solutions

→ **Decide** — Choose the best candidate according to the strategy

→ **Improve** — Move to a better solution if found

→ **Stop** — Stop when the stopping criterion is met

Repeat the cycle until the stopping criterion is satisfied.

TYPICAL SEARCH PROCESS (ITERATIVE)

1 INITIALIZE

- Generate an initial feasible solution.
- Set algorithm parameters.

→ **2 GENERATE NEIGHBORS**

- Create neighboring solutions from the current solution.
- Use move operators or construction rules.

→ **3 EVALUATE SOLUTIONS**

- Check feasibility (constraints).
- Compute objective value (fitness).

→ **4 SELECT THE BEST**

- Choose the best candidate according to the selection strategy.
- Accept or reject the candidate.

→ **5 MOVE TO NEW SOLUTION**

- Update the current solution.
- Update the best solution found.

→ **6 CHECK STOPPING CRITERION**

- Time limit reached?
- Max iterations?
- No improvement for several iterations?
- Target objective achieved?

No → Continue searching

Yes → Stop

COMMON SEARCH STRATEGIES

Local Search
Improve the current solution step by step.

Constructive Heuristics
Build a feasible solution from scratch.

Metaheuristics
Use higher-level strategies to guide the search (e.g., GA, SA, TS, ACO).

GOAL OF THE SEARCH

Find the best possible solution within the available time and computational resources.

Not always optimal, but good enough!

KEY TAKEAWAYS

- The search process explores many solutions intelligently.
- It balances exploration (finding new areas) and exploitation (improving current solutions).
- Heuristics make the search feasible and efficient.
- A good search process leads to better solutions.

6. ILLUSTRATIVE OPTIMIZATION PROBLEM

Interpreting heuristic solutions



Heuristic solutions are not always optimal, but they are good enough within reasonable time and resources.
Interpreting the results helps us understand their quality, strengths, limitations, and practical usefulness.

WHAT DOES INTERPRETING MEAN?

1. EVALUATE QUALITY
Assess how good the solution is compared to the goal, constraints, and available benchmarks.

2. UNDERSTAND TRADE-OFFS
Identify the trade-offs made (e.g., cost vs. quality, time vs. accuracy).

3. ANALYZE PATTERNS
Look for patterns, insights, or reasons behind the solution structure.

4. JUDGE PRACTICALITY
Determine if the solution is practical, implementable, and aligned with real-world needs.

EXAMPLE: STORE LOCATION PROBLEM (HEURISTIC SOLUTION)

Heuristic Result



- 5 store locations selected using a greedy heuristic
- Total expected profit: $1,250,000
- Computation time: 2 minutes
- Best known solution (benchmark): $1,320,000 (found by exact method)

INTERPRETATION

1. **Quality:** The solution achieves 94.7% of the best known profit ($1,250,000 / $1,320,000). This is close to optimal.
2. **Trade-offs:** The heuristic prioritized high-profit areas, but some low-cost locations with moderate profit were not chosen.
3. **Patterns:** Selected locations are concentrated near high-demand zones and major roads.
4. **Practicality:** The solution is easy to implement and fits the budget and time constraints. It is a strong practical choice.

KEY TAKEAWAY

A heuristic solution may not be perfect, but interpreting it helps us:

- Understand its value
- Know its limitations
- Make informed decisions
- Improve future searches

TIPS FOR INTERPRETING HEURISTIC SOLUTIONS

- Always compare with a baseline or benchmark.
- Consider multiple criteria, not just one objective.
- Visualize results to gain better insights.
- Involve domain experts to validate practicality and feasibility.
- Use insights to refine heuristics and improve future solutions.

TASK

1 FORM A GROUP

Form a group consisting of **5 students**.



2 DOWNLOAD A RESEARCH ARTICLE

Download a scientific journal article that discusses an optimization problem solved using a heuristic technique, published **no later than 2020**.



3 WRITE A BRIEF REVIEW

Write a brief review of the selected article in **one page**.



