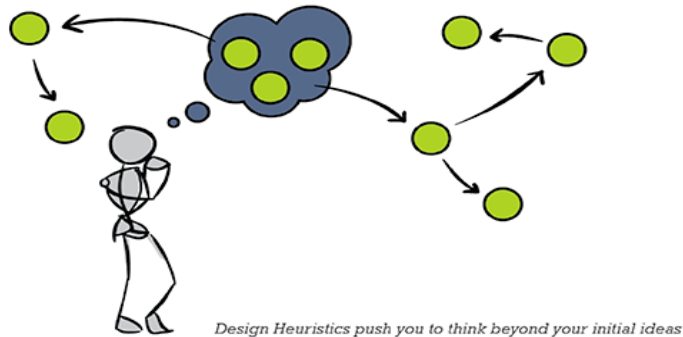


Heuristics Algorithm Design



Dr. Bibhudatta Sahoo

Communication & Computing Group

CS215, Department of CSE, NIT Rourkela

Email: bdsahu@nitrkl.ac.in, 9937324437, 2462358

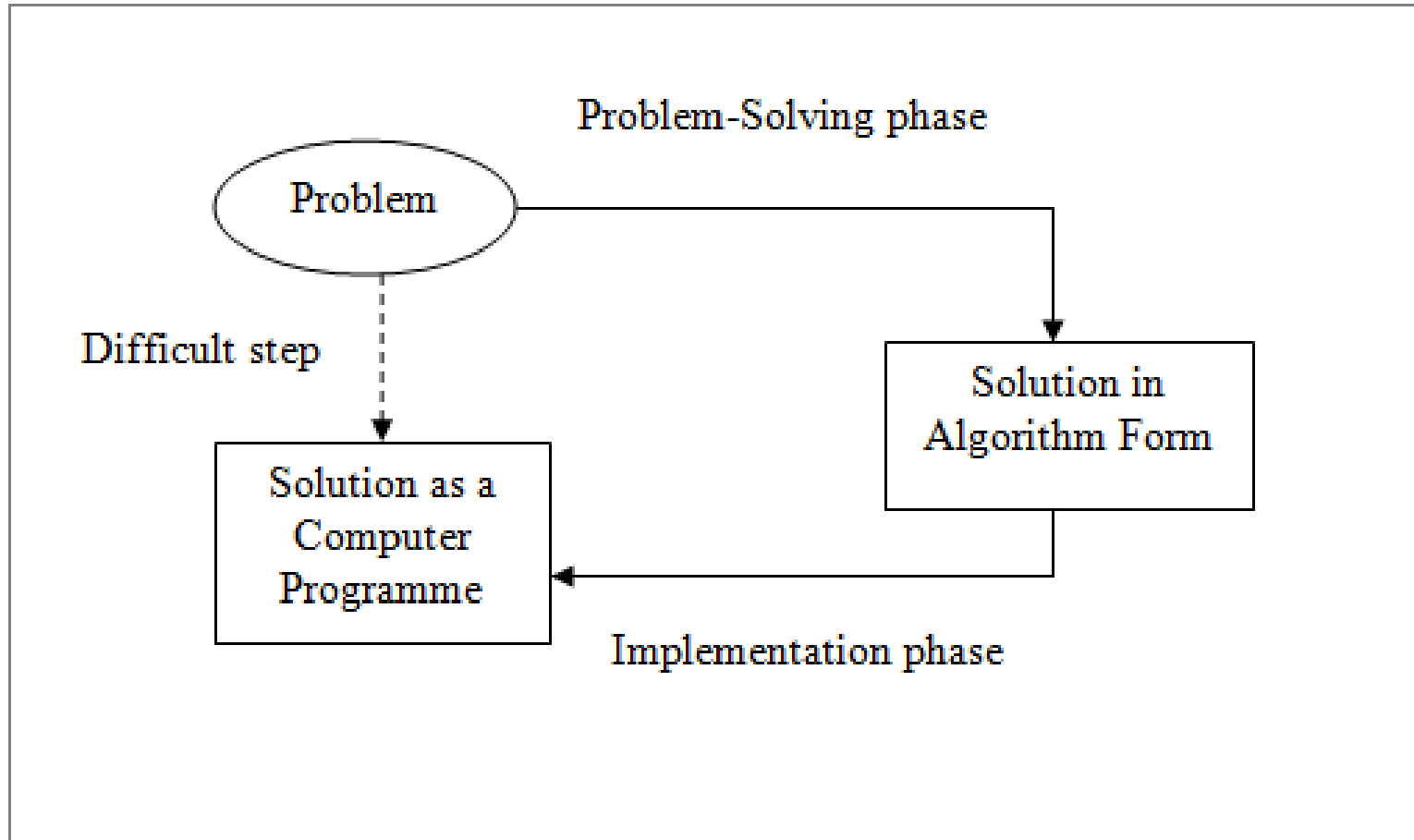
Optimization Problem

Mathematically speaking, optimization can be described as follows: Given a function $f : S \rightarrow \mathbb{R}$ which is called the objective function, find the argument which minimizes f :

$$x^* = \arg \min_{x \in S} f(x)$$

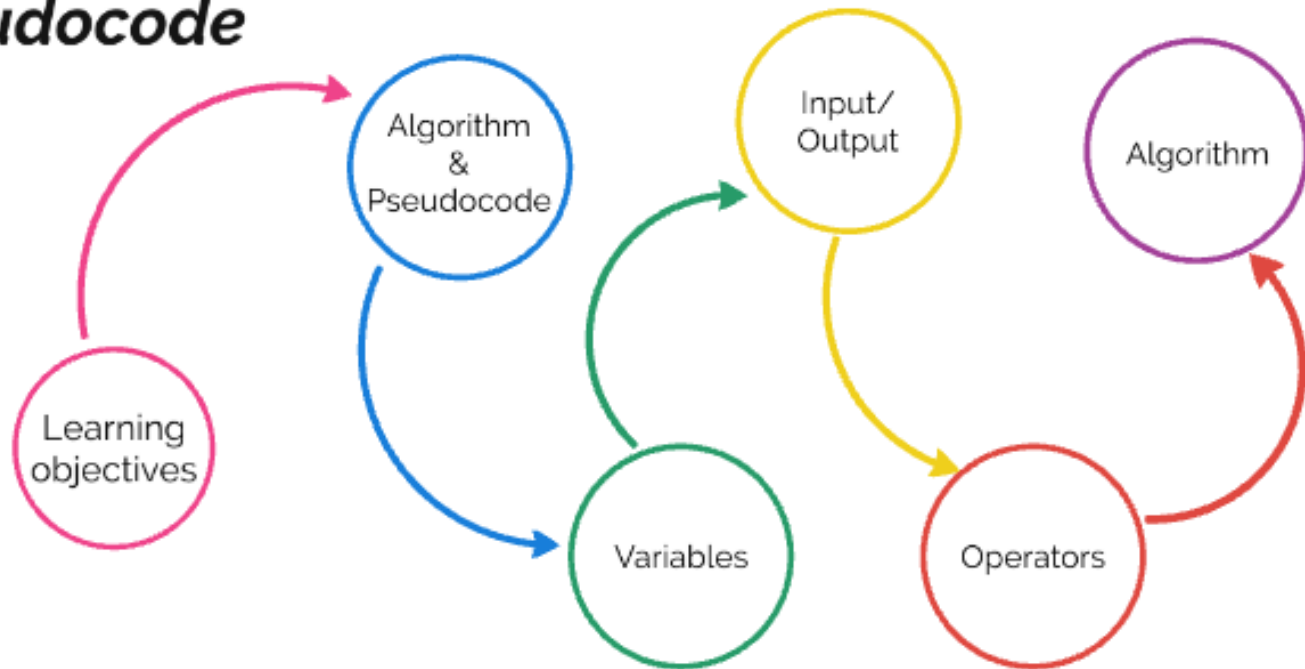
S defines the so-called solution set, which is the set of all possible solutions for the optimization problem. Sometimes, the unknown(s) x are referred to design variables. The function f describes the optimization criterion, i.e., enables us to calculate a quantity which indicates the “quality” of a particular x .

Problem Solving process

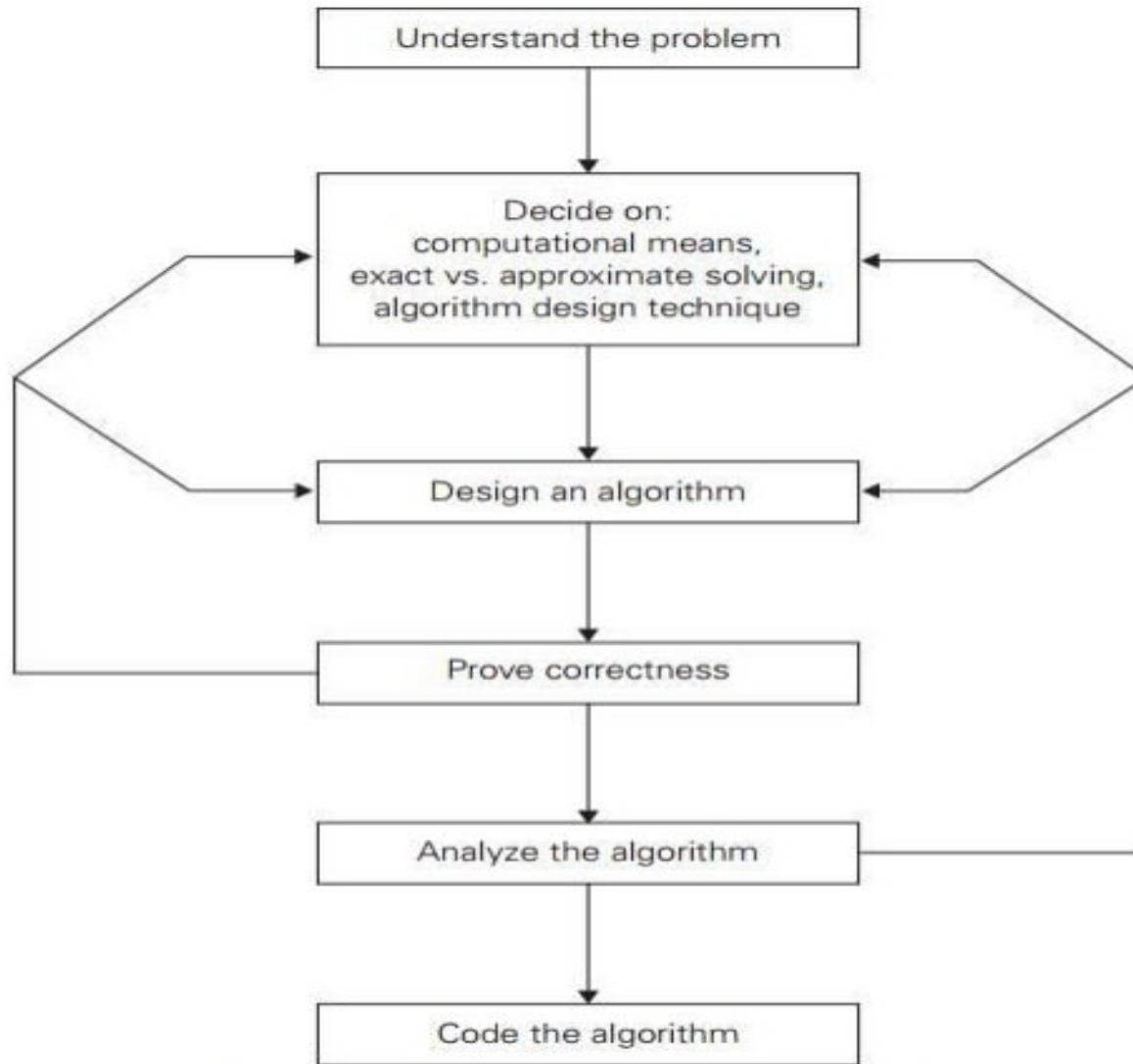



Algorithm development

Algorithm design and problem solving *Pseudocode*

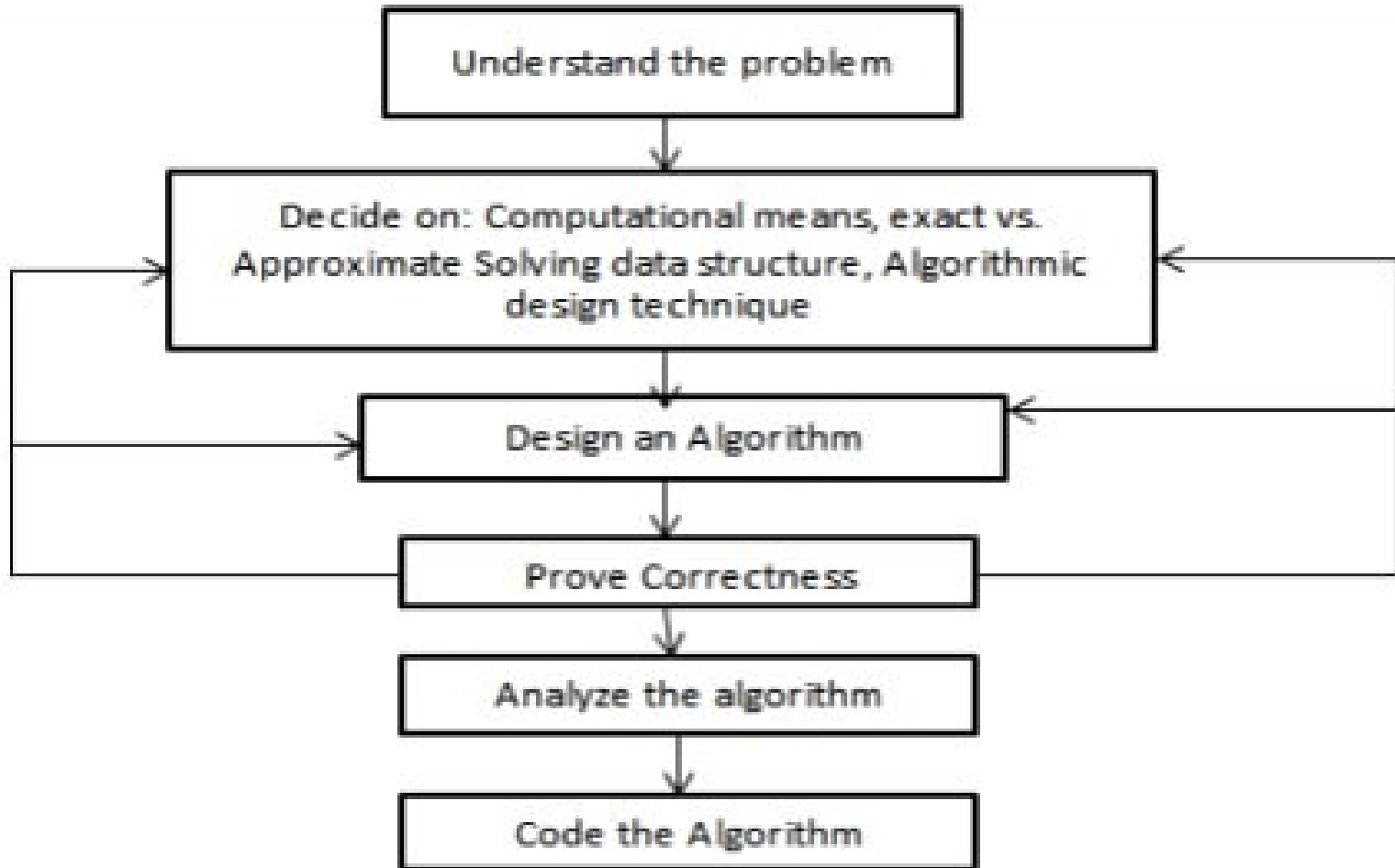


Algorithm Design Analysis Process



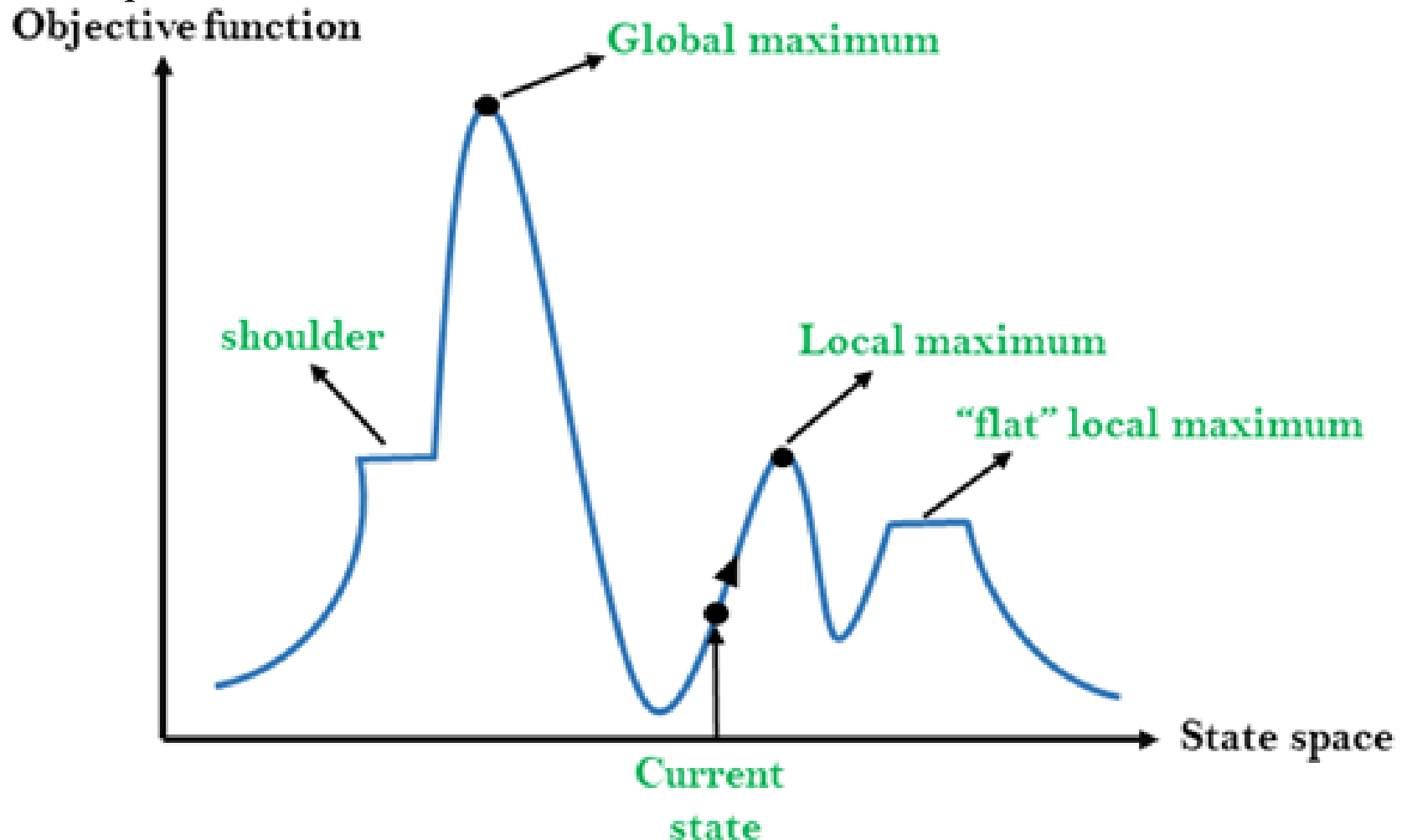
 Algorithm design and analysis process.

Algorithmic problem solving



A simple state-space diagram

The objective function has been shown on the y-axis, while the state-space represents the x-axis.



Various regions of a simple state-space diagram

A state-space diagram provides a graphical representation of states and the optimization function. If the objective function is the y-axis, we aim to establish the local maximum and global maximum.

Various regions

- **Local maximum:** A local maximum is a solution that surpasses other neighboring solutions or states but is not the best possible solution.
- **Global maximum:** This is the best possible solution achieved by the algorithm.
- **Current state:** This is the existing or present state.
- **Flat local maximum:** This is a flat region where the neighboring solutions attain the same value.
- **Shoulder:** This is a plateau whose edge is stretching upwards.

Solution space is exponential

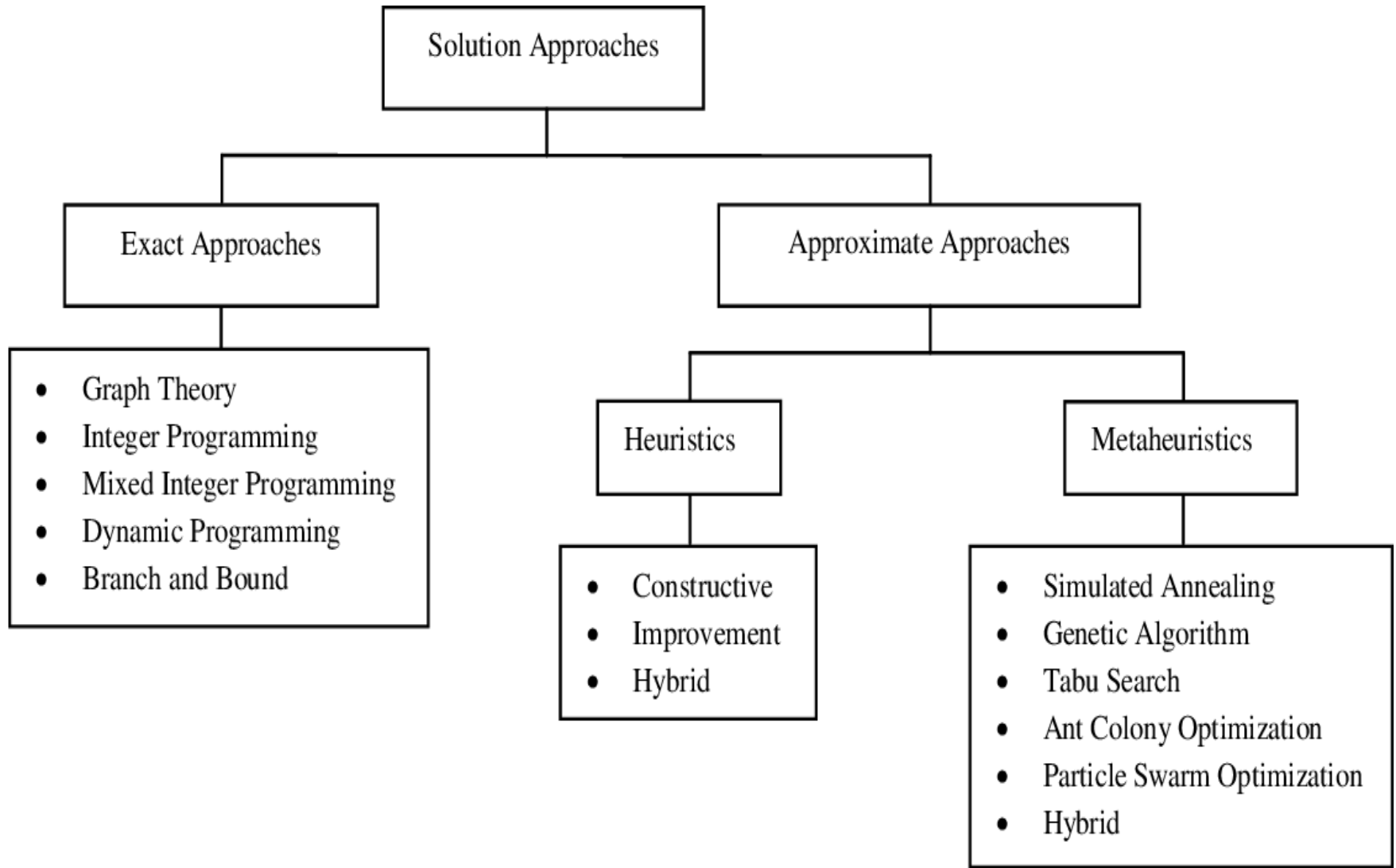
	<i>constant</i>	<i>logarithmic</i>	<i>linear</i>	<i>N-log-N</i>	<i>quadratic</i>	<i>cubic</i>	<i>exponential</i>
<i>n</i>	$O(1)$	$O(\log n)$	$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n^3)$	$O(2^n)$
1	1	1	1	1	1	1	2
2	1	1	2	2	4	8	4
4	1	2	4	8	16	64	16
8	1	3	8	24	64	512	256
16	1	4	16	64	256	4,096	65536
32	1	5	32	160	1,024	32,768	4,294,967,296
64	1	6	64	384	4,069	262,144	1.84×10^{19}

Running times for different sizes of input.

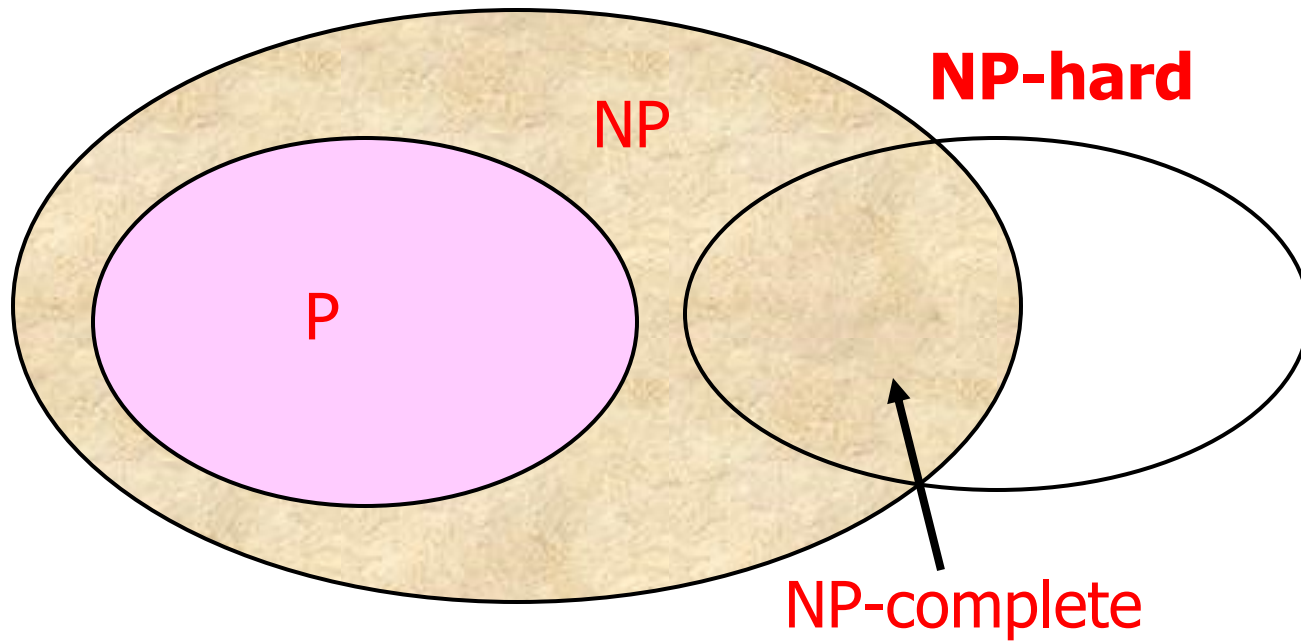
n	$\log n$	n	$n \log n$	n^2	n^3	2^n
8	3 nsec	0.01 μ	0.02 μ	0.06 μ	0.51 μ	0.26 μ
16	4 nsec	0.02 μ	0.06 μ	0.26 μ	4.10 μ	65.5 μ
32	5 nsec	0.03 μ	0.16 μ	1.02 μ	32.7 μ	4.29 sec
64	6 nsec	0.06 μ	0.38 μ	4.10 μ	262 μ	5.85 cent
128	0.01 μ	0.13 μ	0.90 μ	16.38 μ	0.01 sec	10^{20} cent
256	0.01 μ	0.26 μ	2.05 μ	65.54 μ	0.02 sec	10^{58} cent
512	0.01 μ	0.51 μ	4.61 μ	262.14 μ	0.13 sec	10^{135} cent
2048	0.01 μ	2.05 μ	22.53 μ	0.01 sec	1.07 sec	10^{598} cent
4096	0.01 μ	4.10 μ	49.15 μ	0.02 sec	8.40 sec	10^{1214} cent
8192	0.01 μ	8.19 μ	106.50 μ	0.07 sec	1.15 min	10^{2447} cent
16384	0.01 μ	16.38 μ	229.38 μ	0.27 sec	1.22 hrs	10^{4913} cent
32768	0.02 μ	32.77 μ	491.52 μ	1.07 sec	9.77 hrs	10^{9845} cent
65536	0.02 μ	65.54 μ	1048.6 μ	0.07 min	3.3 days	10^{19709} cent
131072	0.02 μ	131.07 μ	2228.2 μ	0.29 min	26 days	10^{39438} cent
262144	0.02 μ	262.14 μ	4718.6 μ	1.15 min	7 mnths	10^{78894} cent
524288	0.02 μ	524.29 μ	9961.5 μ	4.58 min	4.6 years	10^{157808} cent
1048576	0.02 μ	1048.60 μ	20972 μ	18.3 min	37 years	10^{315634} cent

Note: “nsec” stands for nanoseconds, “ μ ” is one microsecond and “cent” stands for centuries. The explosive running time (measured in centuries) when it is of the order 2^n .

Searching an solution space



P, NP, NP-Complete and NP-Hard Problems



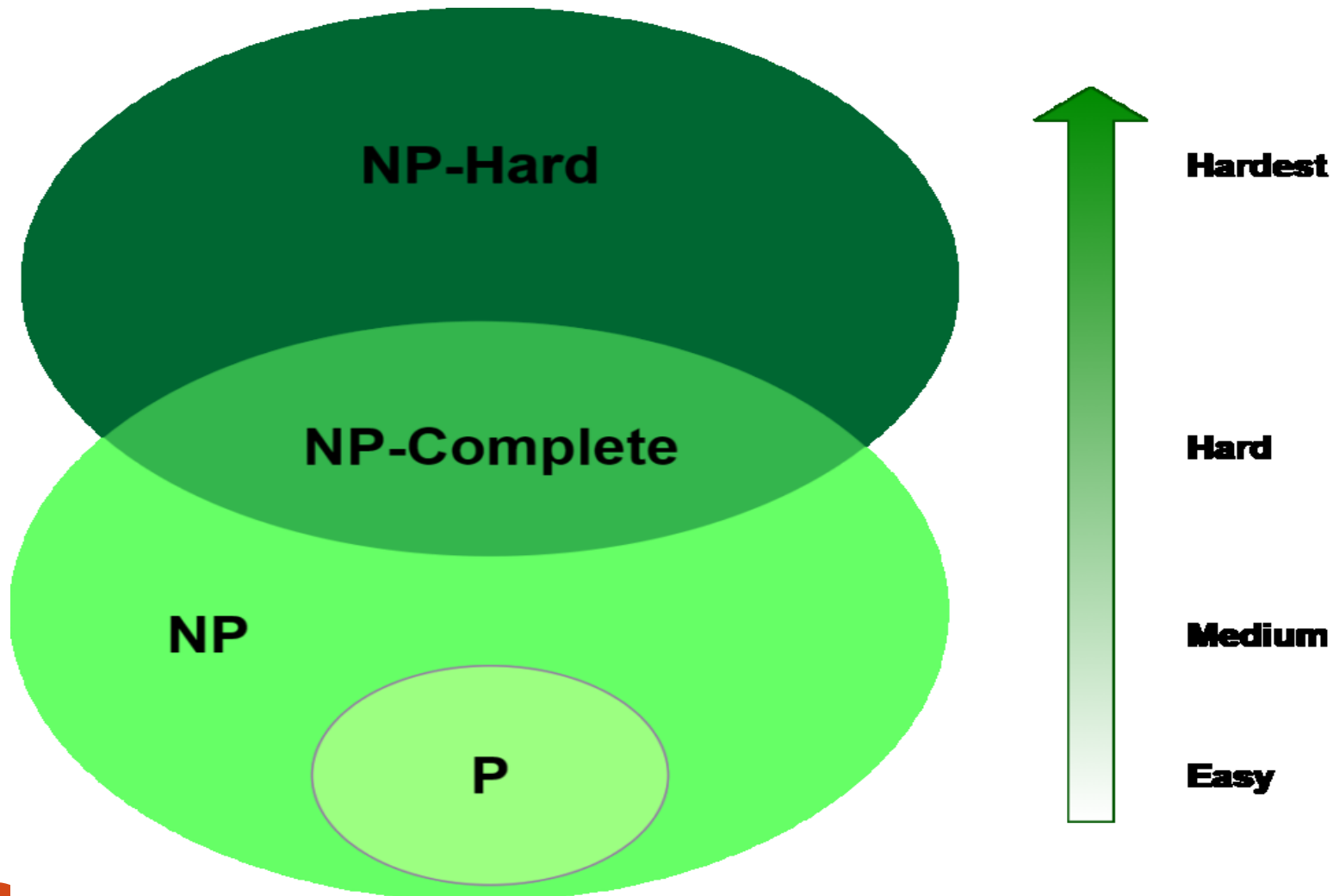
Relevant facts about NP-complete

1. If any NP-complete has a polynomial time algorithm then they all do. Another way to think of this is that all pairs of NP-complete problems are reducible to each other via a reduction that runs in **polynomial time**.
2. There are thousands of known natural NP-complete problems from every possible area of application that people would like to find polynomial time algorithms for.
3. No one knows of a polynomial time algorithm for any NP-complete problem.
4. No one knows of a proof that no NP-complete problem can have a polynomial time algorithm.
5. Almost all computer scientists believe that NP-complete problems do not have **polynomial time algorithms**.

Relevant facts about NP-complete

6. If we can find a polynomial time algorithm for an NP-Complete problem, then all problems in NP have polynomial time algorithms.
7. If we can prove that one problem in NP can't be solved in polynomial time, then all NP-Complete problems are not solvable in polynomial time.
8. All NP-Complete problems appear to be difficult.
9. We are not aware of any polynomial time algorithms to solve them.
10. However, there is no proof that polynomial time algorithms do not exist!

P, NP, NP-Complete and NP-Hard Problems in Computer Science



What is the definition of P, NP, NP-complete and NP-hard?

	P	NP	NP- complete	NP-hard
Solvable in polynomial time	✓			
Solution verifiable in polynomial time	✓	✓	✓	
Reduces any NP problem in polynomial time			✓	✓

Methods adopted to cope with NP-complete Problem

- Use dynamic programming, backtracking or branch-and-bound technique to reduce the computational cost. This might work if the problem size is not too big.
- Find the sub-problem of the original problem that have polynomial time solution.
- Use **approximation algorithms** to find approximate solution in polynomial time.
- Use **randomized algorithm** to find solutions in affordable time with a high probability of correctness of the solution.
- Use **heuristic** like greedy method, simulated annealing or genetic algorithms etc. However, the solutions produced cannot be guaranteed to be within a certain distance from the optimal solution.

Heuristics in Computer Science

- Heuristics in computer science and artificial intelligence are “rules of thumb” used in algorithms to assist in finding approximate solutions to complex problems.
- Often, there’s simply too much data to sift through in order to come to a solution in a timely manner, so a heuristic algorithm is used to trade exactness for speed.
- Because heuristics are based on individual rules unique to the problem they are solving, the specifics of the heuristics vary from problem to problem.
- Heuristics aim to produce solutions in a *reasonable time frame* that are *good enough* for solving the problem at hand.
- The solution produced using a heuristic may not be the perfect or exact solution, but it’s valuable as an approximate or best-guess solution.
- Some problems would require hundreds of thousands of years for an exact answer, but we can produce an approximate solution almost instantly.

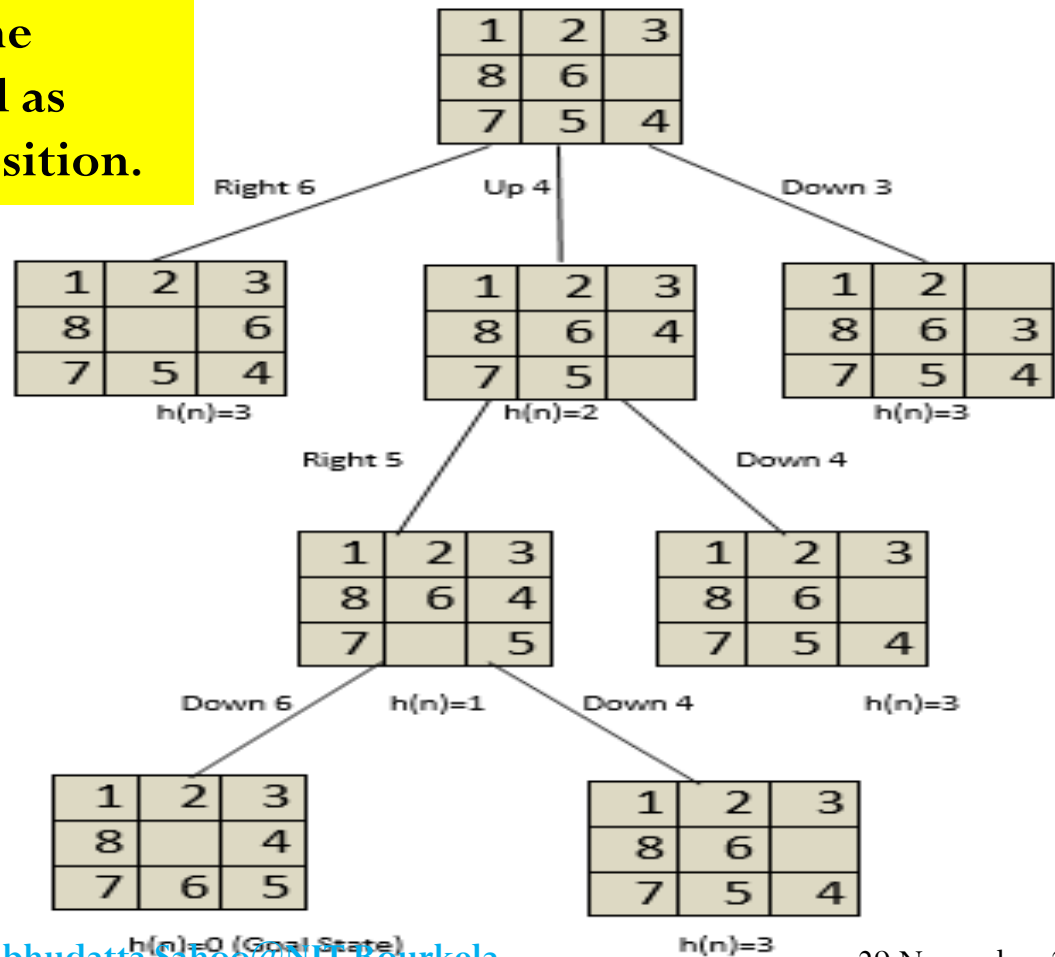
Heuristic Trade-Offs

- The entire value proposition of a heuristic is based on trade-offs. Typically we are trading accuracy for time. That said, there are several different levers we have to pull when designing a good heuristic.
- **Optimality:** Many problems have multiple solutions, for example, “what is a good path to get from city A to city B? Do we need the best path, or will a good path be good enough?
- **Completeness:** When there are multiple valid solutions to a problem do we need to find all of them? Will a subset of valid solutions suffice?
- **Accuracy:** Many questions don’t have a correct answer. For example, “Will Tommy like a pair of boots or a pair of gloves for Christmas?” A heuristic can improve accuracy in these situations.
- **Execution time:** The primary goal of a heuristic is to provide a quick answer that’s good enough. Some heuristics are only marginally quicker than classic methods.

A heuristic function

- A heuristic function, also simply called a heuristic, is a function that ranks alternatives in search algorithms at each branching step based on available information to decide which branch to follow.

A heuristic function for the 8-puzzle problem is defined as $h(n)$ = Number of tiles out of position.



A heuristic function

- A heuristic function for the 8-puzzle problem is **$h(n)$ =Number of tiles out of position.**
- So, there is total of three tiles out of position i.e., 6,5 and 4.
- Do not count the empty tile present in the goal state). i.e. $h(n)=3$. Now, we require to minimize the value of **$h(n) = 0$.**
- It is seen from the state space tree that the goal state is minimized from $h(n)=3$ to $h(n)=0$. However, we can create and use several heuristic functions as per the requirement.
- It is also clear from the above example that a heuristic function $h(n)$ can be defined as the information required to solve a given problem more efficiently.
- The information can be related to the nature of the state, cost of transforming from one state to another, goal node characteristics, etc., which is expressed as a heuristic function

Properties of a Heuristic search Algorithm

Use of heuristic function in a heuristic search algorithm leads to following properties of a heuristic search algorithm:

- **Admissible Condition:** An algorithm is said to be admissible, if it returns an optimal solution.
- **Completeness:** An algorithm is said to be complete, if it terminates with a solution (if the solution exists).
- **Dominance Property:** If there are two admissible heuristic algorithms **A1** and **A2** having **h1** and **h2** heuristic functions, then **A1** is said to dominate **A2** if **h1** is better than **h2** for all the values of node **n**.
- **Optimality Property:** If an algorithm is **complete**, **admissible**, and **dominating** other algorithms, it will be the best one and will definitely give an optimal solution.

Heuristic Algorithm Design

Heuristic Techniques

Are ways to approach solving a problem

Mental shortcuts in thinking process of problem solving

Solutions are not expected to be 'perfect'

Aid to creative thinking when overcoming a problem

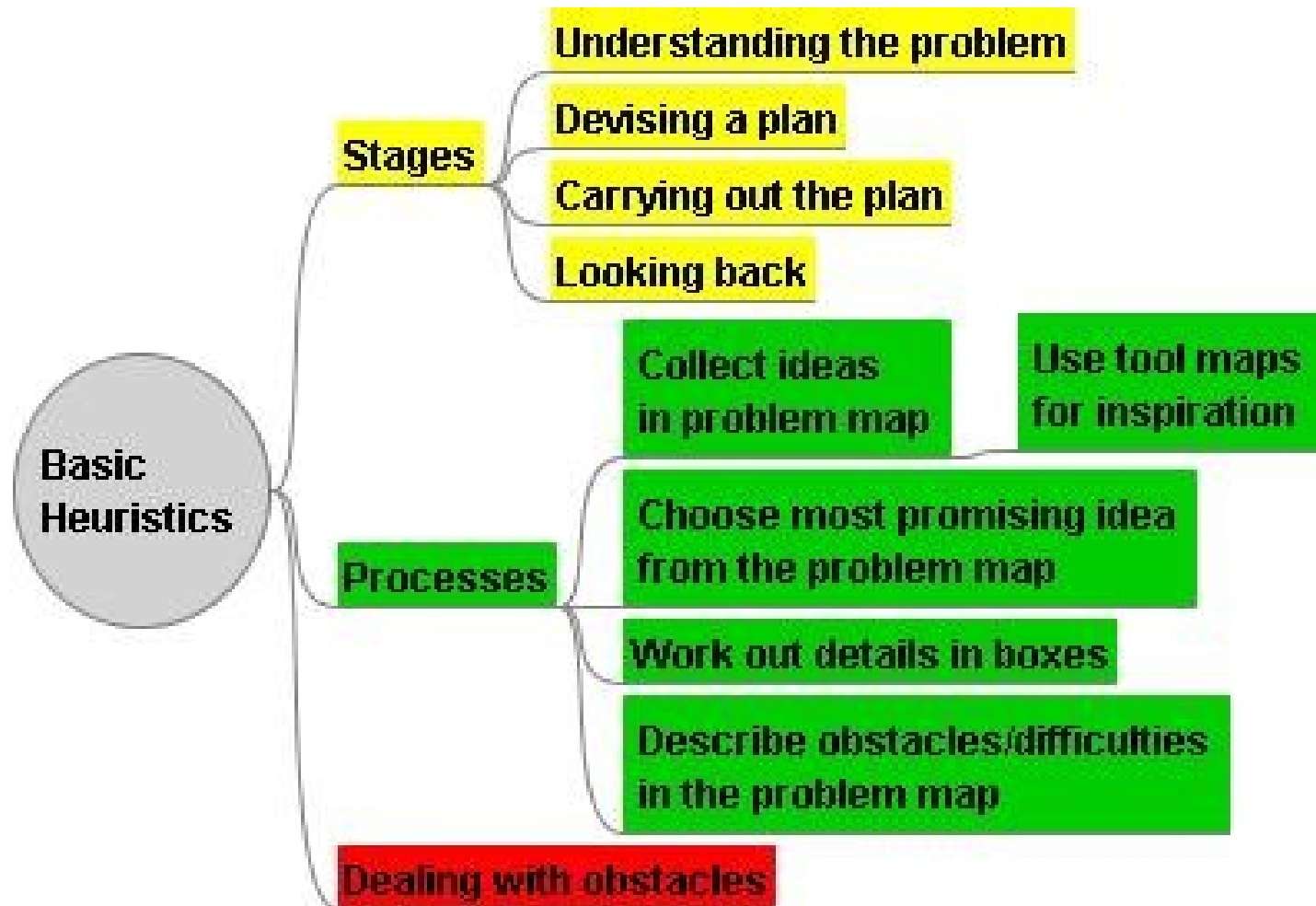
Heuristic algorithm

- A heuristic algorithm is one that is **designed to solve a problem in a faster and more efficient fashion than** traditional methods by **sacrificing optimality**, accuracy, precision, or completeness for speed.
- Heuristic algorithms often times used to solve **NP-complete problems**, a class of decision problems.
- A heuristic, or a heuristic technique, is **any approach to problem-solving that uses a practical method or various shortcuts in order to produce solutions** that may not be optimal but are sufficient given a limited timeframe or deadline.
- In computer science, a **heuristic algorithm is a problem solving method** that uses incomplete information to derive a potentially inaccurate or imprecise solution.
- Heuristics are intended to gain computational performance or conceptual simplicity, potentially at the cost of accuracy or precision

Heuristic algorithm

- The term **heuristic** is used for algorithms which find solutions among all possible ones ,but they do not guarantee that the best will be found, therefore they may be considered as **approximately** and not accurate algorithms.
- **Heuristic** algorithms, usually find a solution close to the best one and they find it **fast** and **easily**.
- Sometimes these algorithms can be accurate, that is they actually find the best solution, but the algorithm is still called heuristic until this best solution is proven to be the best.
- The method used from a heuristic algorithm is one of the known methods, such as greediness, but in order to be easy and fast the algorithm ignores or even suppresses some of the problem's demands.

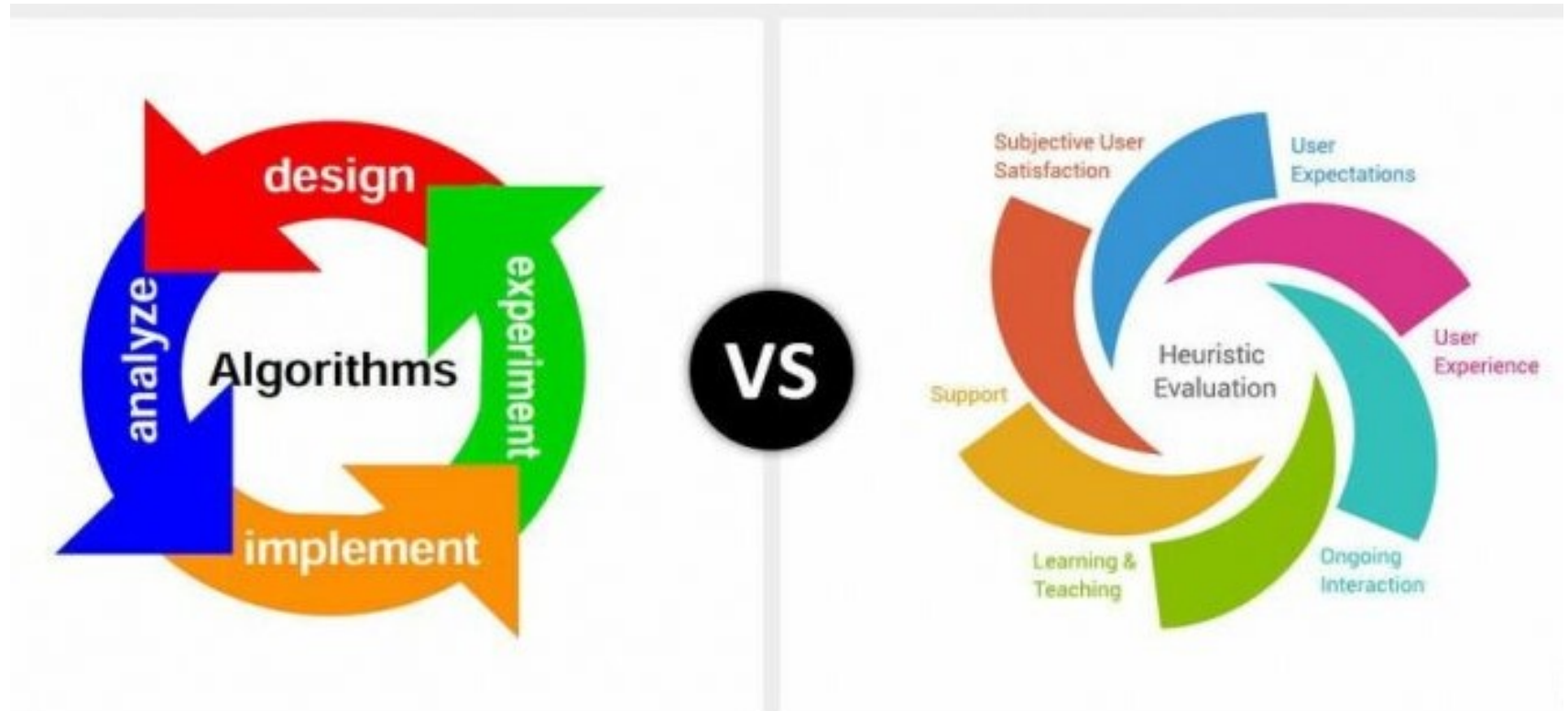
Heuristic algorithm



Fundamentals of Heuristic Algorithm

- It's used to design the solutions to the problems as quickly as possible. **It may not produce the best solution, but it'll give a near-optimal solution in a short time.**
- Heuristic algorithms are used to solve **NP problems** and decrease the time complexity of problems by giving quick solutions.
- In the heuristic algorithm, a **heuristic function** gives the **heuristic value** to find the optimal solution.
- Each node has a heuristic value that is used to find the optimal path:

Algorithmic vs. Heuristic Work



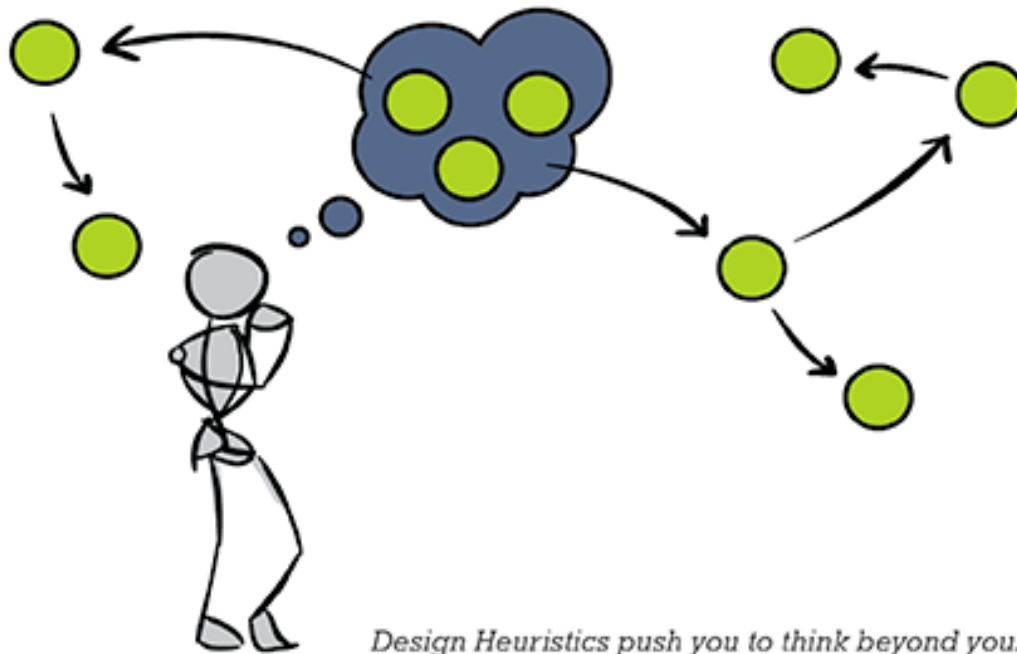
Algorithm vs Heuristic

- An **algorithm** is a step-wise procedure for solving a specific problem **in a** finite number of steps. The result (output) of an **algorithm** is predictable and reproducible given the same parameters (input).
- A **heuristic** is an educated guess which serves as a guide for subsequent explorations. Unlike an algorithm, the results of a heuristic are neither **predictable** nor **reproducible**.
- A real-world comparison of algorithms and heuristics can be seen in human learning. When a child learns to walk, for example, its approach is heuristic, trying different muscle movements until it finds some that work. But, once **mastered**, walking becomes algorithmic, literally a **set of steps which (should) always produce the desired results**.

@ Source: https://www.bioinformatics.org/wiki/Comparison_of_algorithms_and_heuristics

Algorithm vs Heuristic

- In Algorithmic work the process is defined and the end product is expected.
- Heuristic work is the opposite, because there is no algorithm for it. We devise ideas and strategies, experiment and create hypotheses until a solution is found.

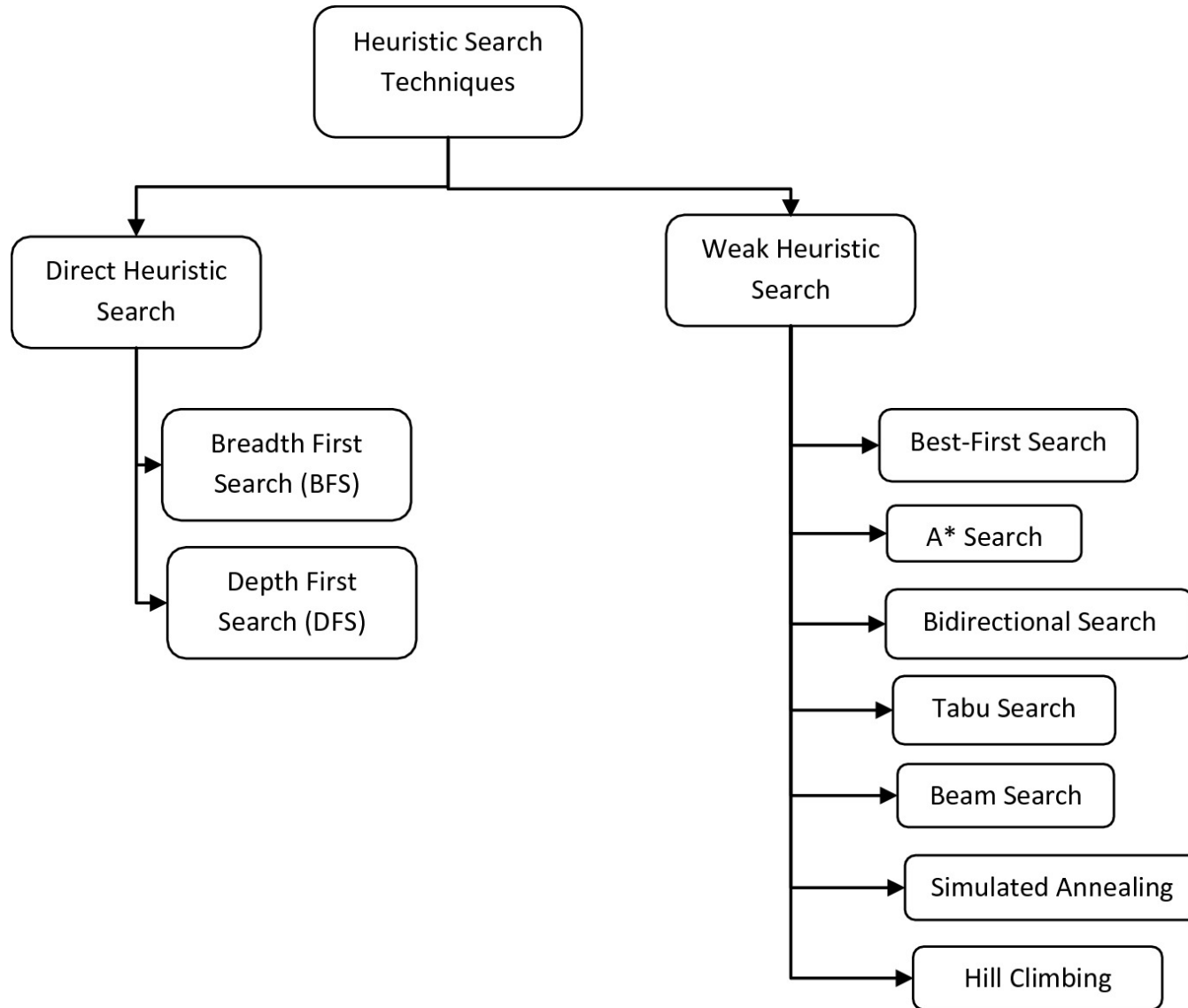


Design Heuristics push you to think beyond your initial ideas

Key points Heuristic

- Heuristics are usually mental shortcuts that help with the thinking processes in problem solving.
- They include using: A rule of thumb, an educated guess, an intuitive judgment, stereotyping, profiling, and common sense.
- Heuristics do not aim for novel solutions, but to implement the known, readily accessible, and loosely applicable.
- The advantage of heuristics is that they can rapidly solve a problem, but it may often result in an immediate, but 'temporary fix' as they are prone to bias.
- Heuristics should be used with care, and as a trigger for formal problem solving, to develop a better, longer term solution.

Heuristic search technique

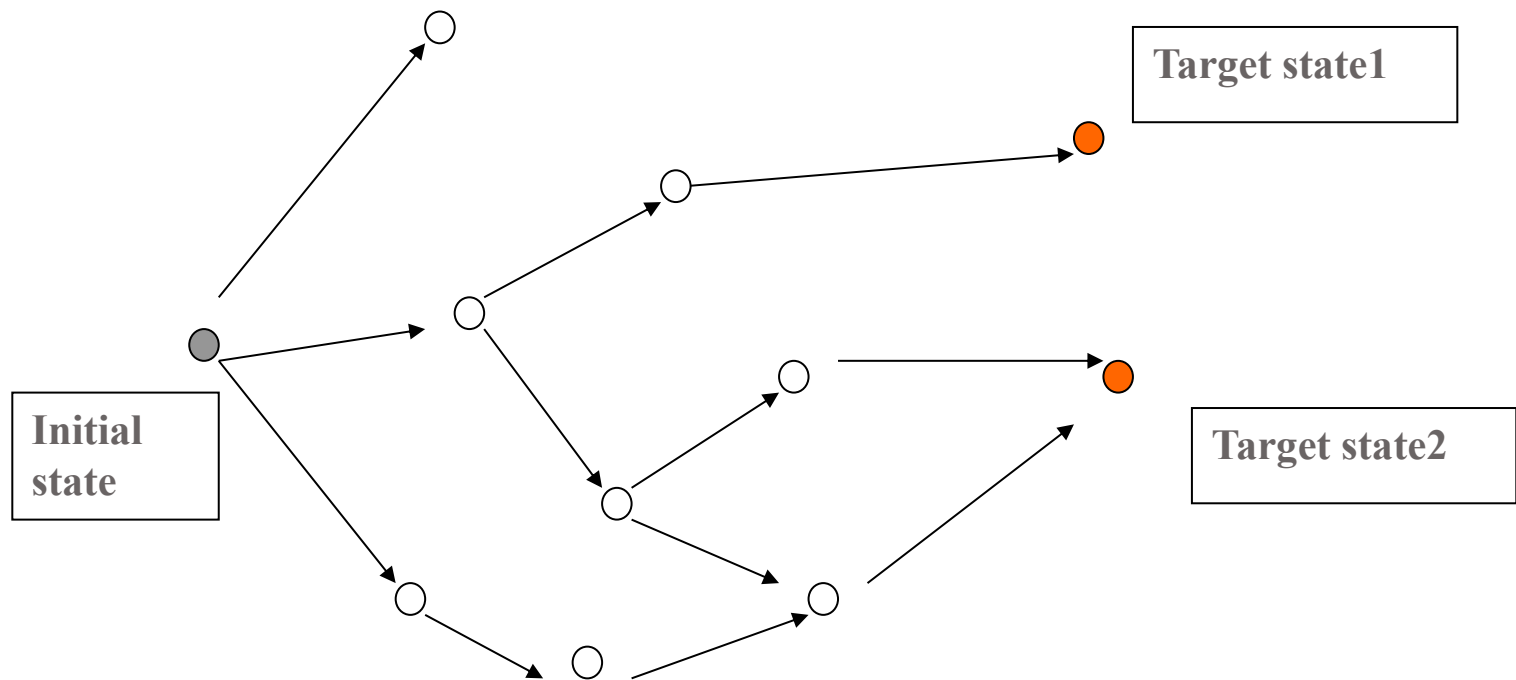


What Are Heuristics?

- A heuristic, or a heuristic technique, is any approach to problem-solving that uses a practical method or various shortcuts in order to produce solutions that may not be optimal but are sufficient given a limited timeframe or deadline.
- Heuristics methods are intended to be flexible and are used for quick decisions, especially when finding an optimal solution is either impossible or impractical and when working with complex data.
- Heuristic is problem-dependent solution strategy where Meta-heuristic is problem-independent solution strategy.

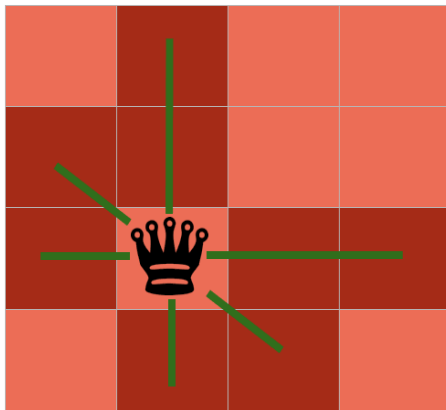
How to solve a problem by searching

- Search in a space of possible state



N-Queens Problem

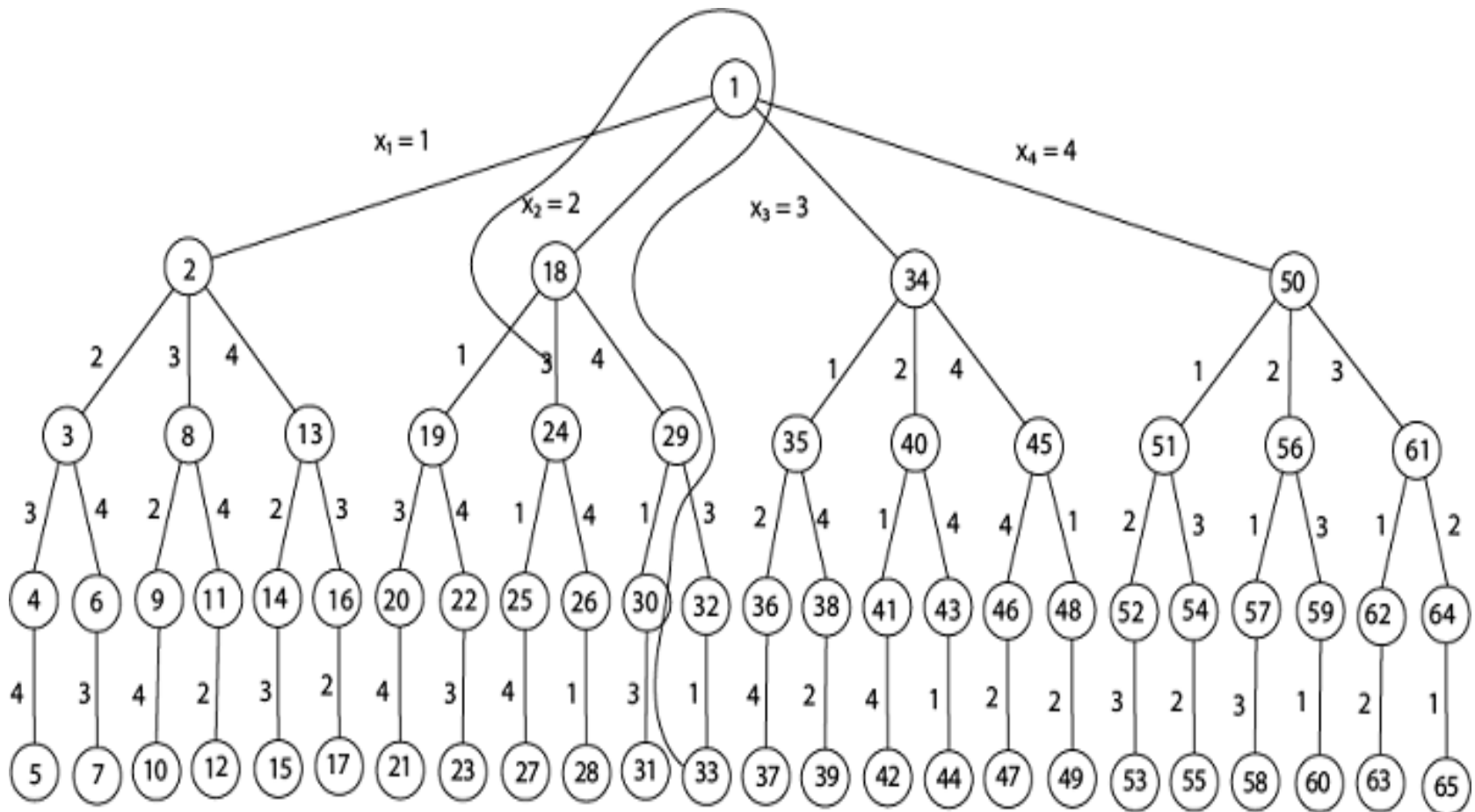
- **N - Queens problem** is to place **n - queens** in such a manner on an $n \times n$ chessboard that no **queens** attack each other by being in the same row, column or diagonal.
- It can be seen that for $n = 1$, the **problem** has a trivial solution, and no solution exists for $n = 2$ and $n = 3$. So first we will consider the 4 queens problem and then generate it to n - queens problem.
- Given a 4×4 chessboard and number the rows and column of the chessboard 1 through 4.



Cells attacked by the queen

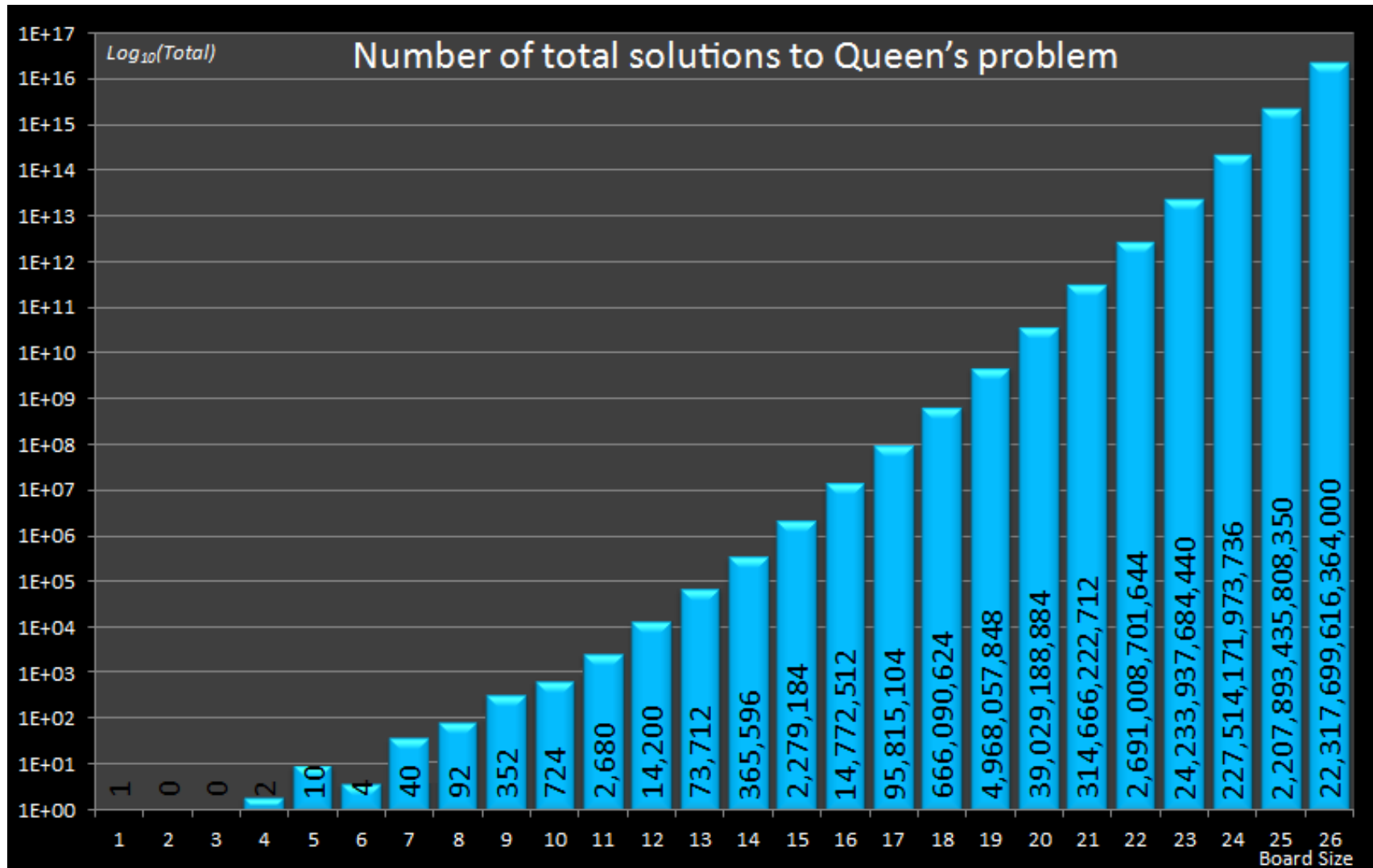
	1	2	3	4
1			q ₁	
2	q ₂			
3				q ₃
4		q ₄		

4- - Queens solution space with nodes numbered in DFS

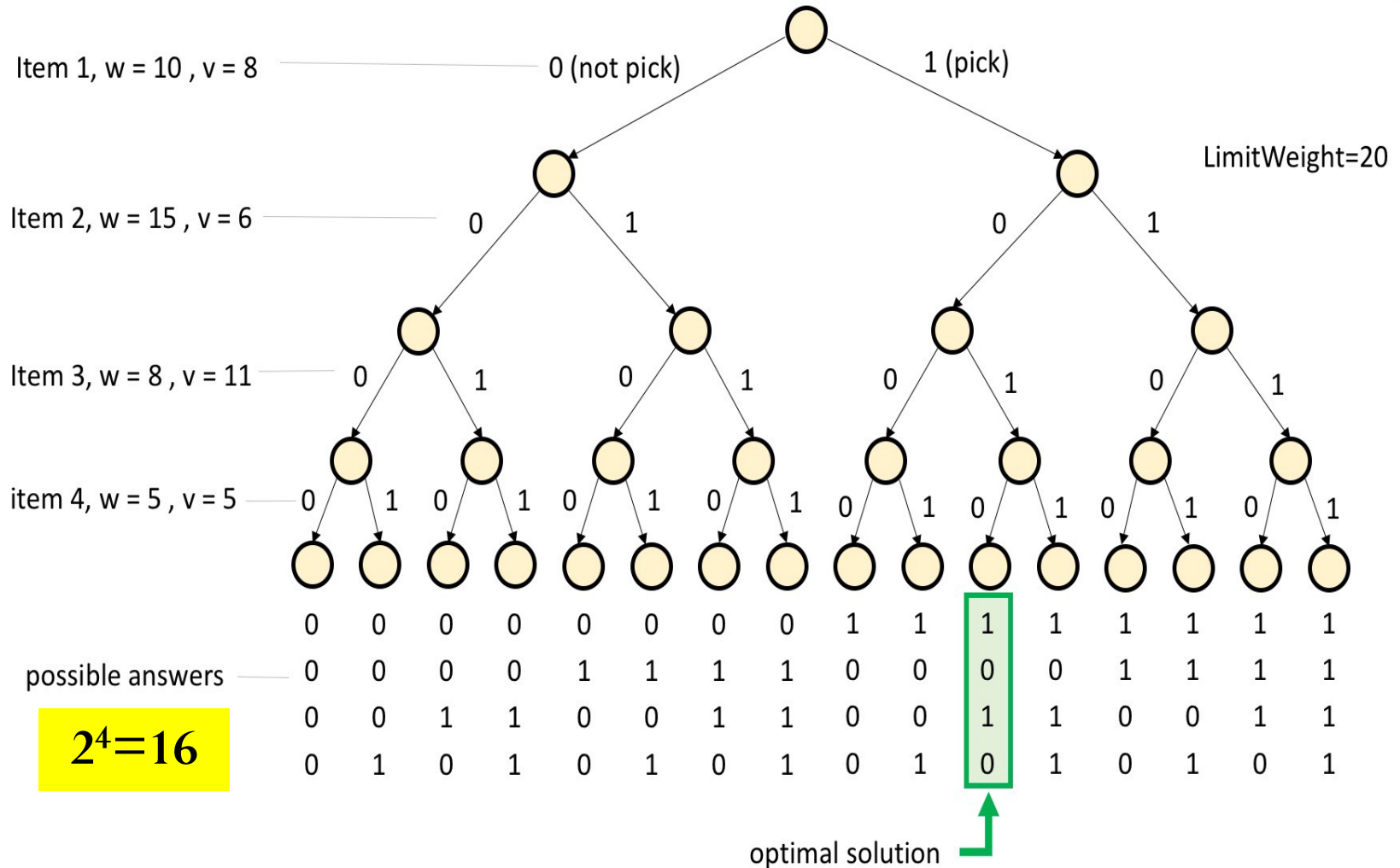


all the solutions to the 4 queens problem can be represented as 4 - tuples (x_1, x_2, x_3, x_4) where x_i represents the column on which queen " q_i " is placed.

Number of total solutions to Queen's problem



Tree organization of solution space : 4 item 0/1 Knapsack problem



Generate and Test

- **Take a state**
- Generate next states by applying relevant rules
(not all rules are applicable for a given state!)
- **Test to see if the goal state is reached.**

If yes - stop

If no - add the generated states into a stack/queue (do not add if the state has already been generated) and repeat the procedure.

- **To get the solution:** Record the paths

Q			

let us first consider an empty chessboard and start by placing the first queen on cell `chessboard[0][0]`

Q			
		Q	

`chessboard[1][2]` is the only possible position where the second queen can be placed

Q			
		Q	
X	X	X	X

no queen can be placed further as queen 1 is in column 1, queen 2 is diagonally opposite to columns 2 and 3; and column 3 has queen 2 in it. Since number of queens is not 4, this is an infeasible solution and will not be printed

	Q		

the next iteration of our algorithm will begin with the second column and start placing the queens again

	Q		
			Q
Q			

queens 2 and 3 are easily placed onto the chessboard without creating the possibility of attacking one another.

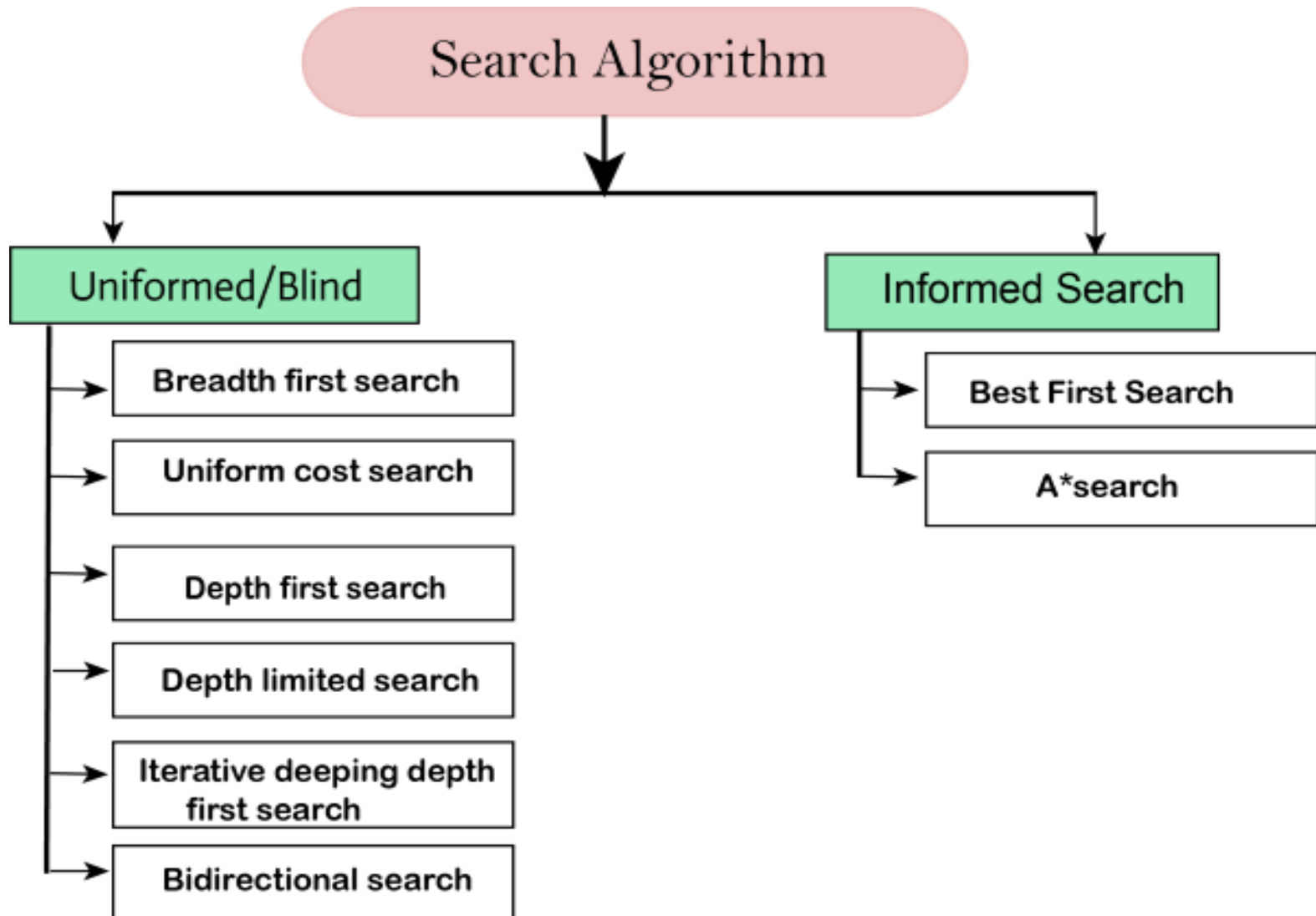
	Q		
			Q
Q			
		Q	

the 4th queen has also been placed accordingly. Since the number of queens = 4, this solution will be printed.

Problem vs. Solution Space

- For a systems developer, the real world may be associated with the **problem space**, the system implementation as the **solution space**.
- The human's way of characterizing the problem or decision space can be called the **problem space**
- The **solution space** is the range of potential solutions that might be recommended. For a knowledge modeler coming from this context, therefore, problem and solution space may both be part of the "real world."
- In optimization a **candidate solution** is a member of a set of possible solutions to a given problem. A candidate solution does not have to be a likely or reasonable solution to the problem.
- The space of all candidate solutions is called the **feasible region**, **feasible set**, **search space**, or **solution space**.

Searching Solution Space



Search Algorithm Terminologies:

- **Search:** Searching is a step by step procedure to solve a search-problem in a given search space. A search problem can have three main factors:
 - **Search Space:** Search space represents a set of possible solutions, which a system may have.
 - **Start State:** It is a state from where agent begins **the search**.
 - **Goal test:** It is a function which observe the current state and returns whether the goal state is achieved or not.
- **Search tree:** A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding to the initial state.
- **Actions:** It gives the description of all the available actions to the agent.
- **Transition model:** A description of what each action do, can be represented as a transition model.
- **Path Cost:** It is a function which assigns a numeric cost to each path.
- **Solution:** It is an action sequence which leads from the start node to the goal node.
- **Optimal Solution:** If a solution has the lowest cost among all solutions.

Properties of Search Algorithms:

- **Completeness:** A search algorithm is said to be complete if it guarantees to return a solution if at least any solution exists for any random input.
- **Optimality:** If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.
- **Time Complexity:** Time complexity is a measure of time for an algorithm to complete its task.
- **Space Complexity:** It is the maximum storage space required at any point during the search, as the complexity of the problem.

Uninformed/Blind Search

- The uninformed search does not contain any domain knowledge such as closeness, the location of the goal.
- It operates in a brute-force way as it only includes information about how to traverse the tree and how to identify leaf and goal nodes.
- Uninformed search applies a way in which search tree is searched without any information about the search space like initial state operators and test for the goal, so it is also called blind search. It examines each node of the tree until it achieves the goal node.

Classification of Blind Search

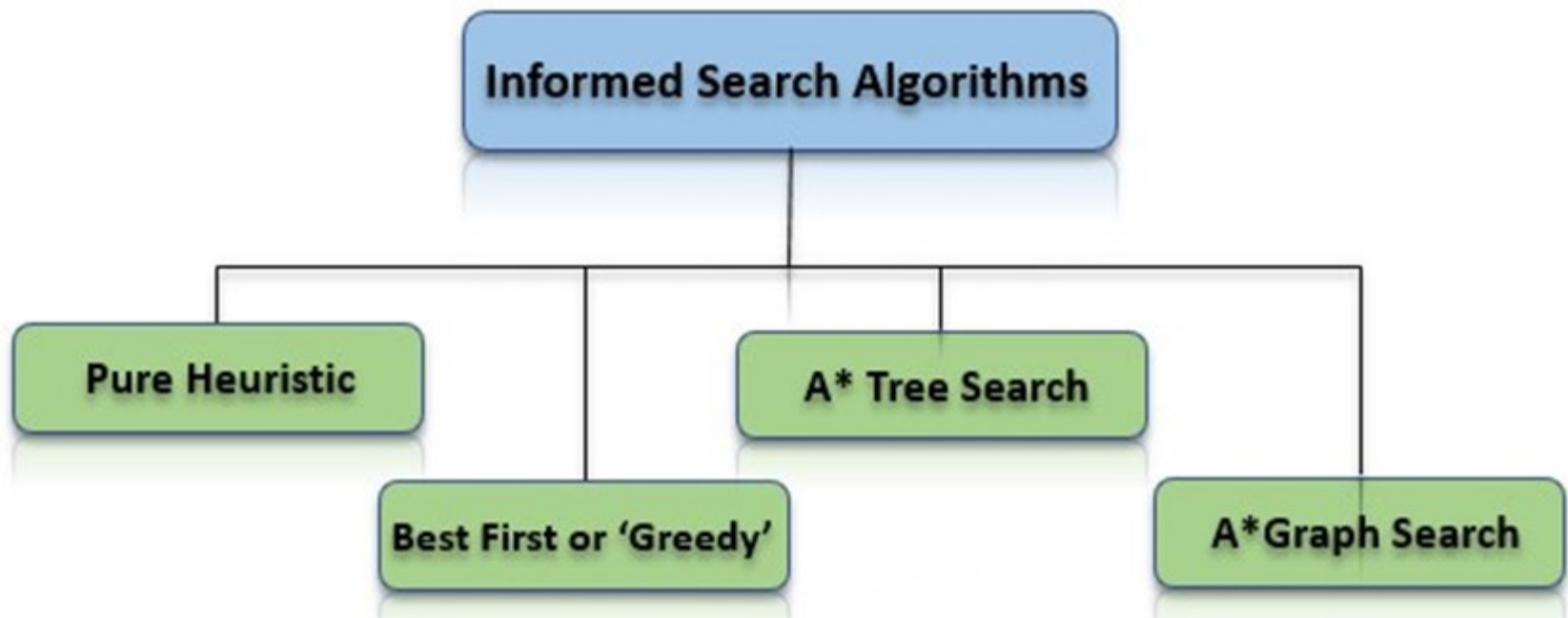
1. Breadth-first search
2. Uniform cost search
3. Depth-first search
4. Iterative deepening depth-first search
5. Bidirectional Search

Informed Search

- Informed search algorithms use **domain knowledge**.
- Informed Search algorithms have information on the **goal state** which helps in more efficient searching. This information is obtained by a function that estimates how close a state is to the goal state.
- In an informed search, problem information is available which can guide the search. Informed search strategies can find a solution more efficiently than an uninformed search strategy.
- Informed search is also called a **Heuristic search**.
- A heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a **good solution** in **reasonable time**.
- **Greedy Search**: Greedy is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. So the problems where choosing locally optimal also leads to global solution are best fit for Greedy.

Informed Search algorithm

- Informed search algorithm contains an array of knowledge such as how far we are from the goal, path cost, how to reach to goal node, etc.
- This knowledge help agents to explore less to the search space and find more efficiently the goal node.



Informed Search: A* Search Algorithm

- A* search is the most widely used informed search algorithm where a node **n** is evaluated by combining values of the functions **g(n)** and **h(n)**.
- A* search algorithm finds the shortest path through the search space using the heuristic function. This search algorithm expands less search tree and provides optimal result faster.
- The function $g(n)$ is the path cost from the start/initial node to a node **n** and $h(n)$ is the estimated cost of the cheapest path from node **n** to the goal node. Therefore, we have

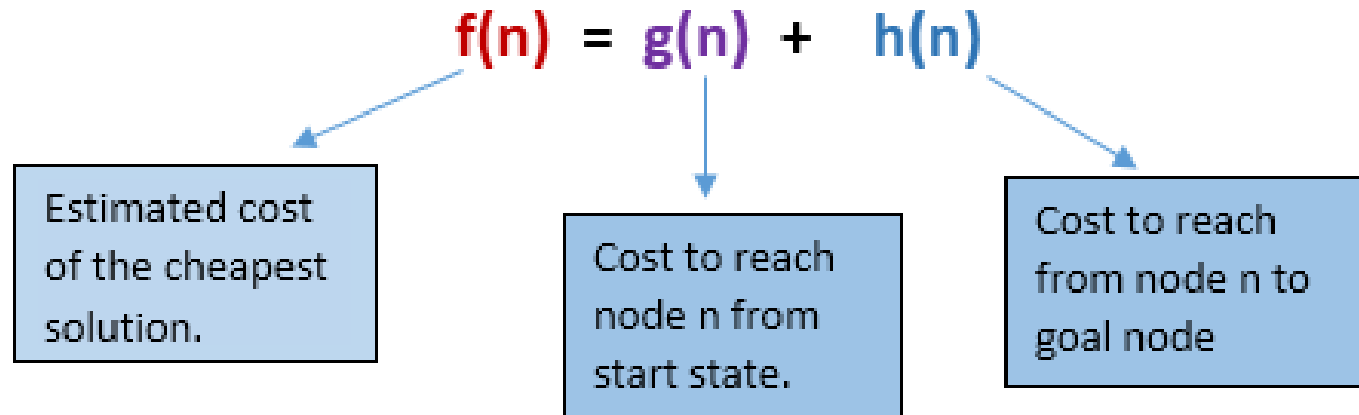
$$f(n) = g(n) + h(n)$$

where $f(n)$ is the estimated cost of the cheapest solution through **n**.

- The optimal solution can be found by finding the lowest values of $f(n)$.

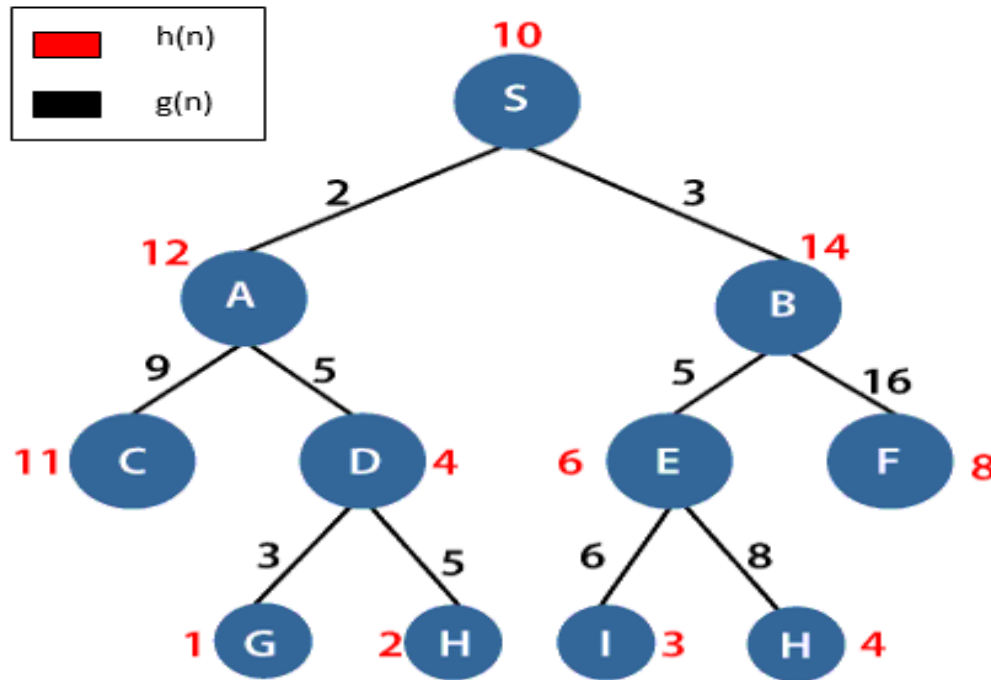
Informed Search: A* Search Algorithm

- In A* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a **fitness number**.



Informed Search: A* Search Algorithm

- S is the root node, and G is the goal node.
- Starting from the root node S and moving towards its next successive nodes A and B.
- In order to reach the goal node G, calculate the $f(n)$ value of node S, A and B using the evaluation equation i.e. $f(n) = g(n) + h(n)$



A*Search

Informed Search: A* Search Algorithm

- **Calculation of $f(n)$ for node S:**

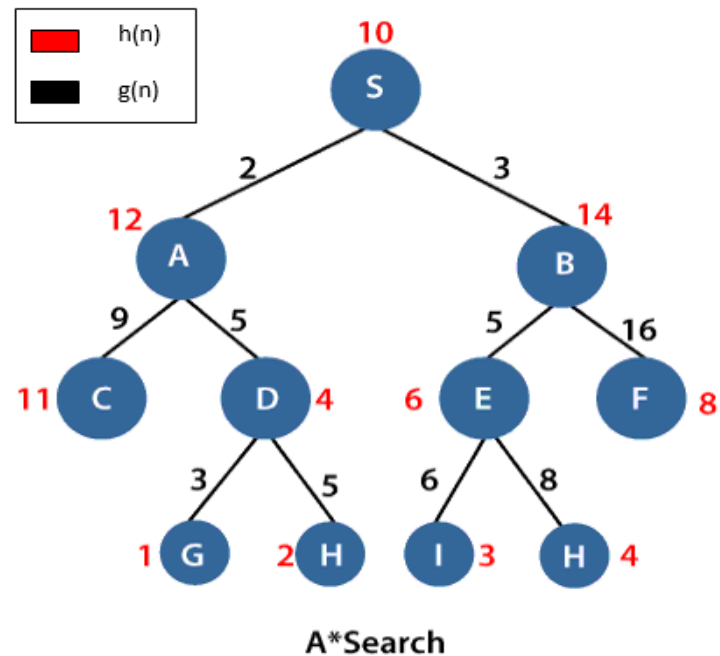
$f(S) = (\text{distance from node S to S}) + h(S)$
 $0 + 10 = 10$.

- **Calculation of $f(n)$ for node A:**

$f(A) = (\text{distance from node S to A}) + h(A)$
 $2 + 12 = 14$

- **Calculation of $f(n)$ for node B:**

$f(B) = (\text{distance from node S to B}) + h(B)$
 $3 + 14 = 17$



Therefore, node A has the lowest $f(n)$ value. Hence, node A will be explored to its next level nodes C and D and again calculate the lowest $f(n)$ value. After calculating, the sequence we get is **S→A→D→G** with **$f(n)=13$ (lowest value)**.

Informed Search

It uses knowledge for the searching process.

It finds solution more quickly.

It may or may not be complete.

Cost is low.

It consumes less time.

It provides the direction regarding the solution.

It is less lengthy while implementation.

Greedy Search, A* Search, Graph Search

Uninformed Search

It doesn't use knowledge for searching process.

It finds solution slow as compared to informed search.

It is always complete.

Cost is high.

It consumes moderate time.

No suggestion is given regarding the solution in it.

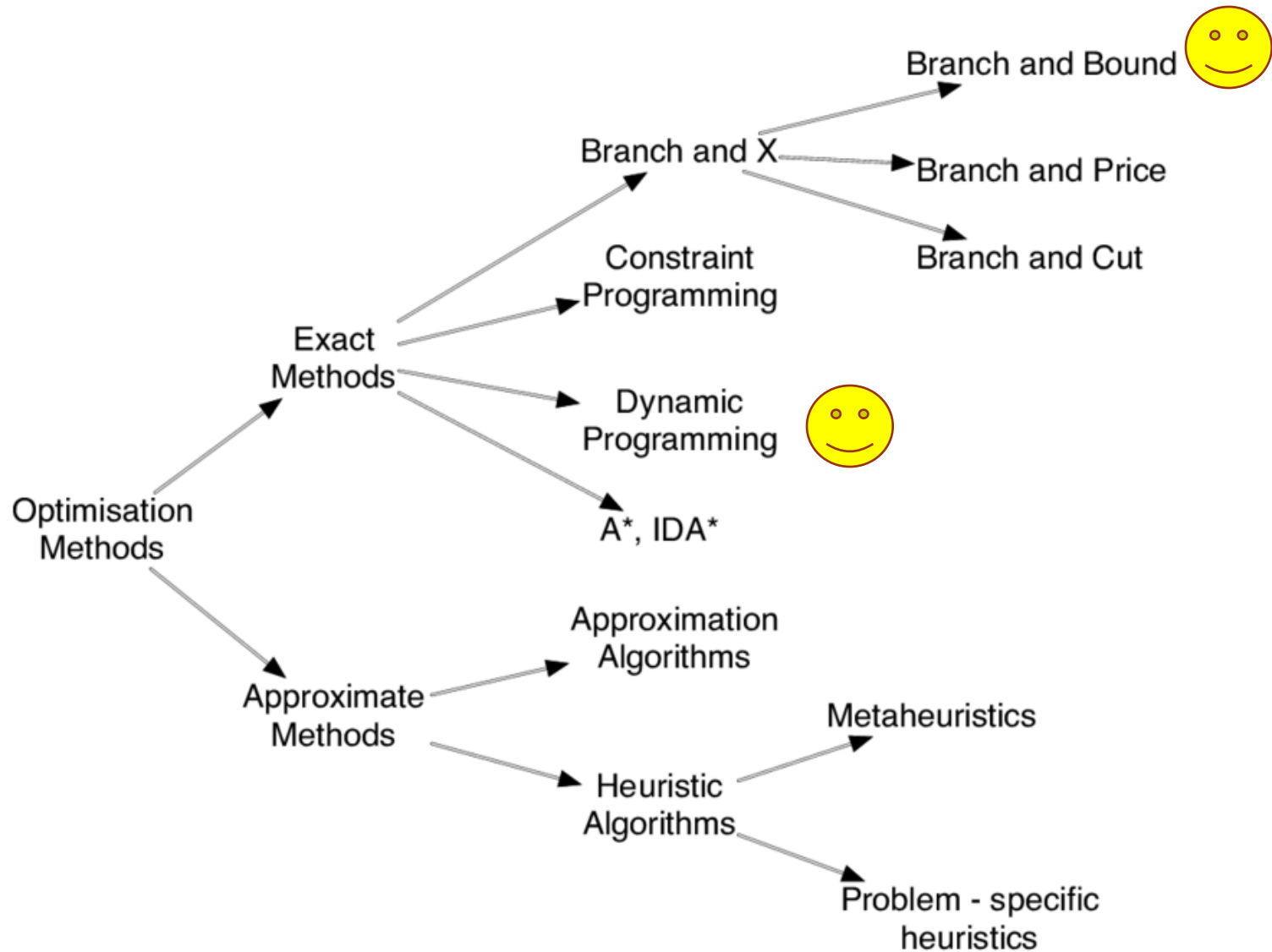
It is more lengthy while implementation.

Depth First Search, Breadth First Search

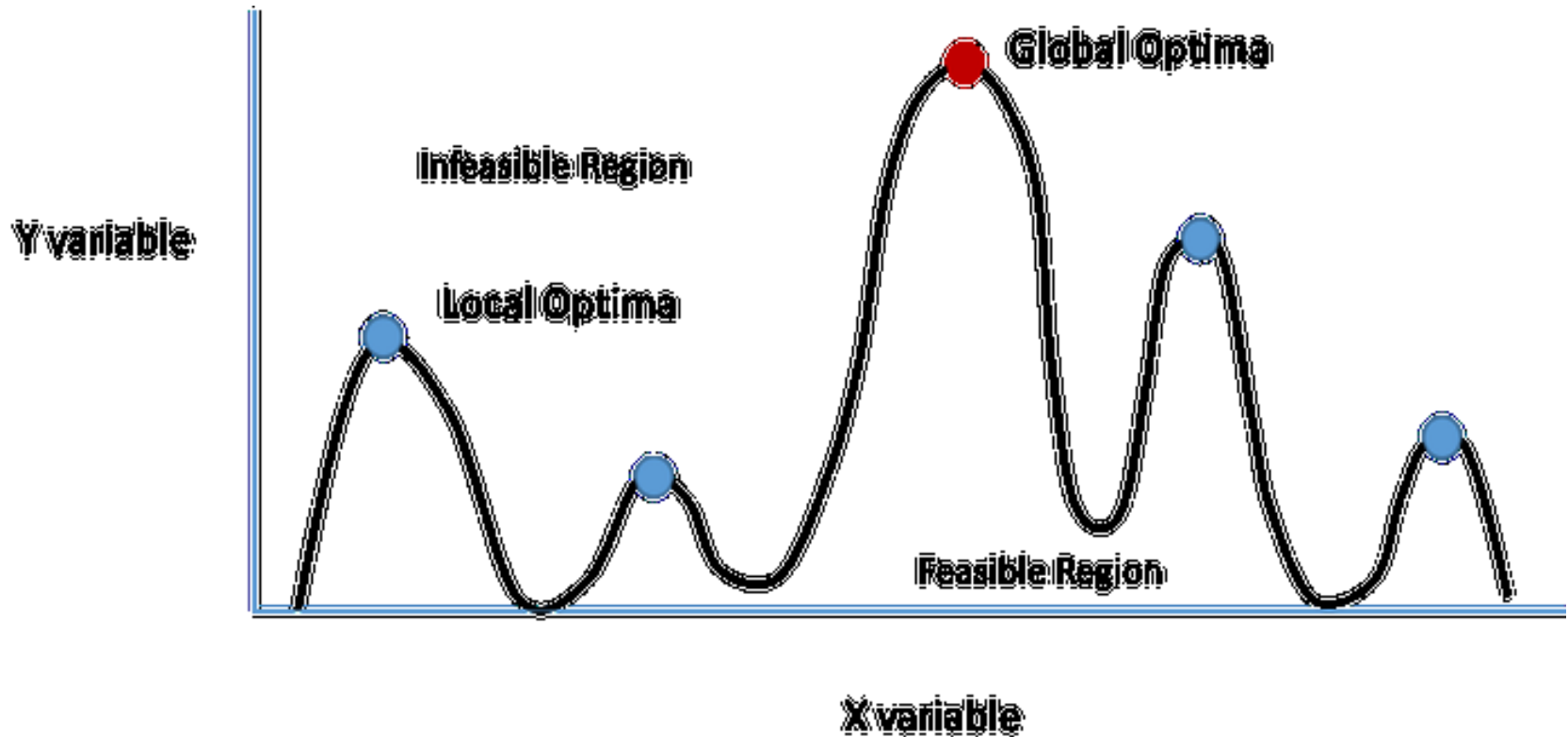
Solving NP-complete problems

- At present, all known algorithms for NP-complete problems require time that is super polynomial in the input size, and it is unknown whether there are any faster algorithms.
- The following techniques can be applied to solve computational problems in general, and they often give rise to substantially faster algorithms:
- **Approximation:** Instead of searching for an optimal solution, search for an "almost" optimal one.
- **Randomization:** Use randomness to get a faster average running time, and allow the algorithm to fail with some small probability.
- **Restriction:** By restricting the structure of the input (e.g., to planar graphs), faster algorithms are usually possible.
- **Parameterization:** Often there are fast algorithms if certain parameters of the input are fixed.
- **Heuristic:** An algorithm that works "reasonably well" on many cases, but for which there is no proof that it is both always fast and always produces a good result. Metaheuristic approaches are often used.

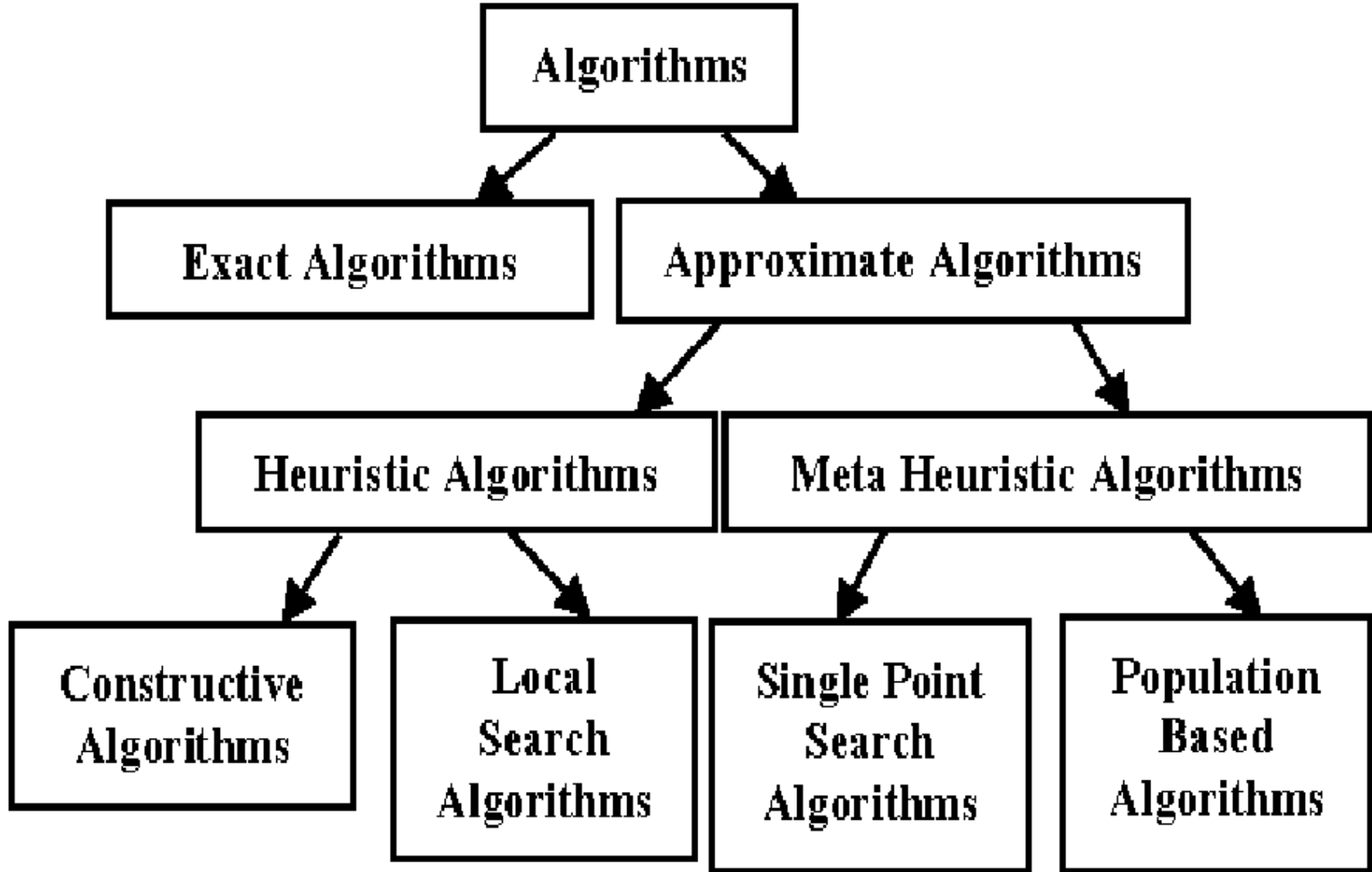
Classification of the Classical Optimizations Methods.



Search space for a optimization problem



Algorithm classification

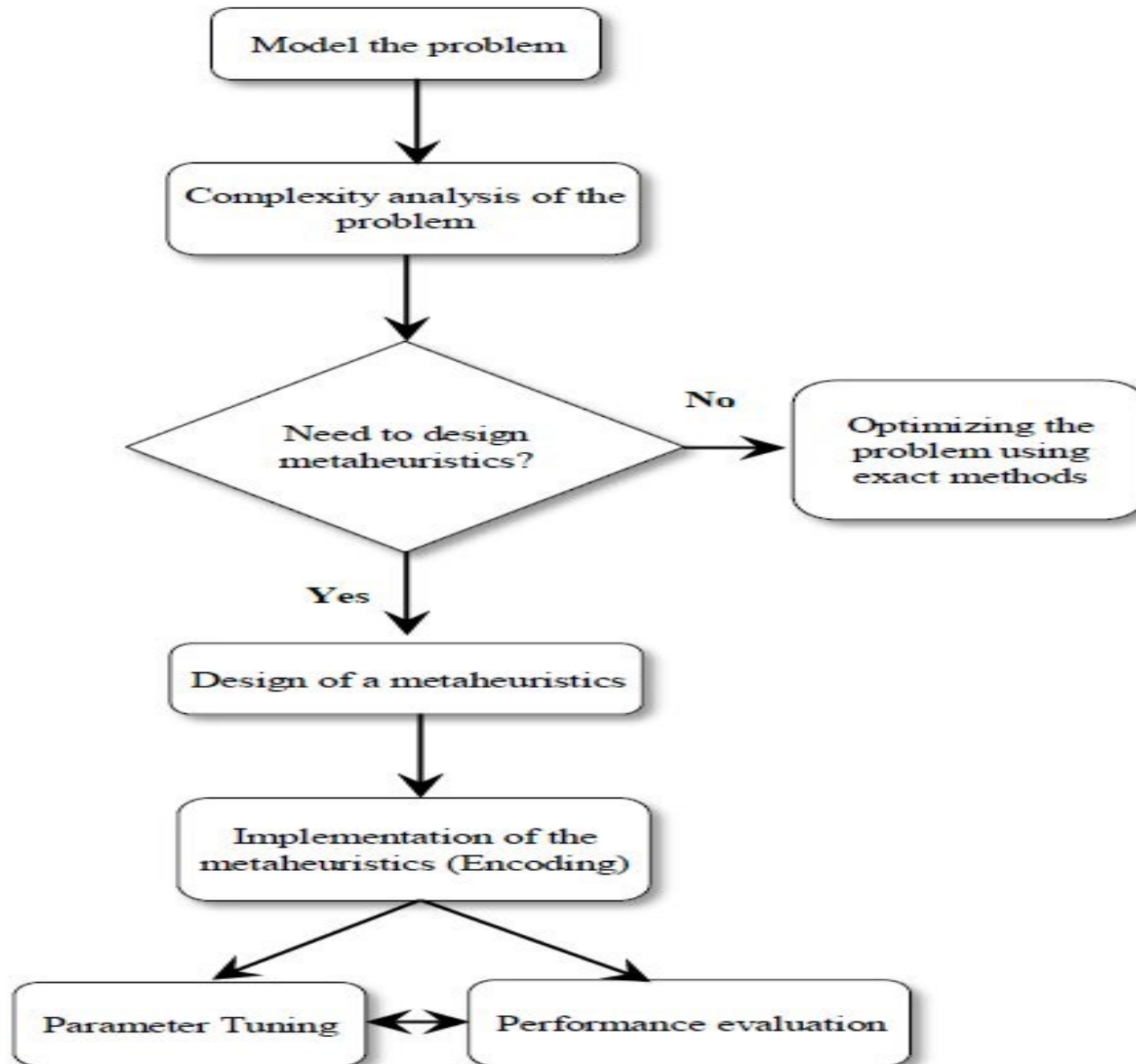


Metaheuristics

Metaheuristics

- Metaheuristics define algorithmic frameworks that can be applied to solve such problems in an approximate way, by combining constructive methods with local and population-based search strategies, as well as strategies for escaping local optima.
- Metaheuristics present a trade-off between exact methods, which may guarantee an optimal solution, although generally at the expense of a huge amount of computer resources, and greedy methods, which are very fast, but usually provide a **low-quality** or **unsatisfactory** solution.
- There is not a single metaheuristic that can provide the best results for any kind of optimization problem, so practitioners should get familiar with the mechanics of metaheuristics and their scope of application.

Metaheuristic's e algorithmic framework



Metaheuristics

- Metaheuristic methods do not guarantee finding the exact optimal solution, but can lead to a near-optimal solution in a computationally efficient manner.
- Most metaheuristic methods are stochastic in nature and mimic a natural, physical or biological principle resembling a search or an optimization process.
- Meta-heuristics are problem-independent optimization techniques which provide an optimal solution by exploring and exploiting the entire search space iteratively.
- These techniques have been successfully engaged to solve distinct real-life and multidisciplinary problems.

Advantages of metaheuristics techniques over the classical optimization methods

1. Metaheuristics can lead to good enough solutions for computationally easy (technically, P class) problems with large input complexity, which can be a hurdle for classical methods.
2. Metaheuristics can lead to good enough solutions for the NP-hard problems, i.e. problems for which no known exact algorithm exists that can solve them in a reasonable amount of time.
3. Unlike most classical methods, metaheuristics require no gradient information and therefore can be used with non-analytic, black-box or simulation-based objective functions.
4. Most metaheuristics have the ability to recover from local optima due to inherent stochasticity or deterministic heuristics specially meant for this purpose.
5. Because of the ability to recover from local optima, metaheuristics can better handle uncertainties in objectives.
6. Most metaheuristics can handle multiple objectives with only a few algorithmic changes.

Metaheuristic algorithms

- **Metaheuristic algorithms** are computational **intelligence paradigms** especially used for sophisticated solving optimization problems.
- In metaheuristic algorithms, meta- means ‘beyond’ or ‘higher level’.
- All metaheuristic algorithms use some tradeoff of local search and global exploration.
- The variety of solutions is often realized via randomization.

Optimization algorithms

- Optimization refers to optimization algorithms that seek the inputs to a function that result in the minimum or maximum of an objective function.
- Optimization algorithms can also be classified as **deterministic** or **stochastic**. If an algorithm works in a mechanical deterministic manner without any random nature, it is called deterministic.
- Stochastic optimization or stochastic search refers to an optimization task that involves **randomness** in some way, such as either from the objective function or in the optimization algorithm.
- A deterministic algorithm may be misled (e.g. “*deceived*” or “*confused*”) by the noisy evaluation of candidate solutions or noisy function gradients, causing the algorithm to bounce around or get stuck (e.g. fail to converge).

Stochastic optimization algorithms

- Randomness in the objective function means that the evaluation of candidate solutions involves some uncertainty or noise and algorithms must be chosen that can make progress in the search in the presence of this noise.
- Randomness in the algorithm is used as a strategy, e.g. stochastic or probabilistic decisions. It is used as an alternative to deterministic decisions in an effort to improve the likelihood of locating the global optima or a better local optima.
- Randomness can help escape local optima and increase the chances of finding a global optimum.
- The randomness used in a stochastic optimization algorithm does not have to be true randomness; instead, pseudorandom is sufficient.
- A pseudorandom number generator is almost universally used in stochastic optimization.

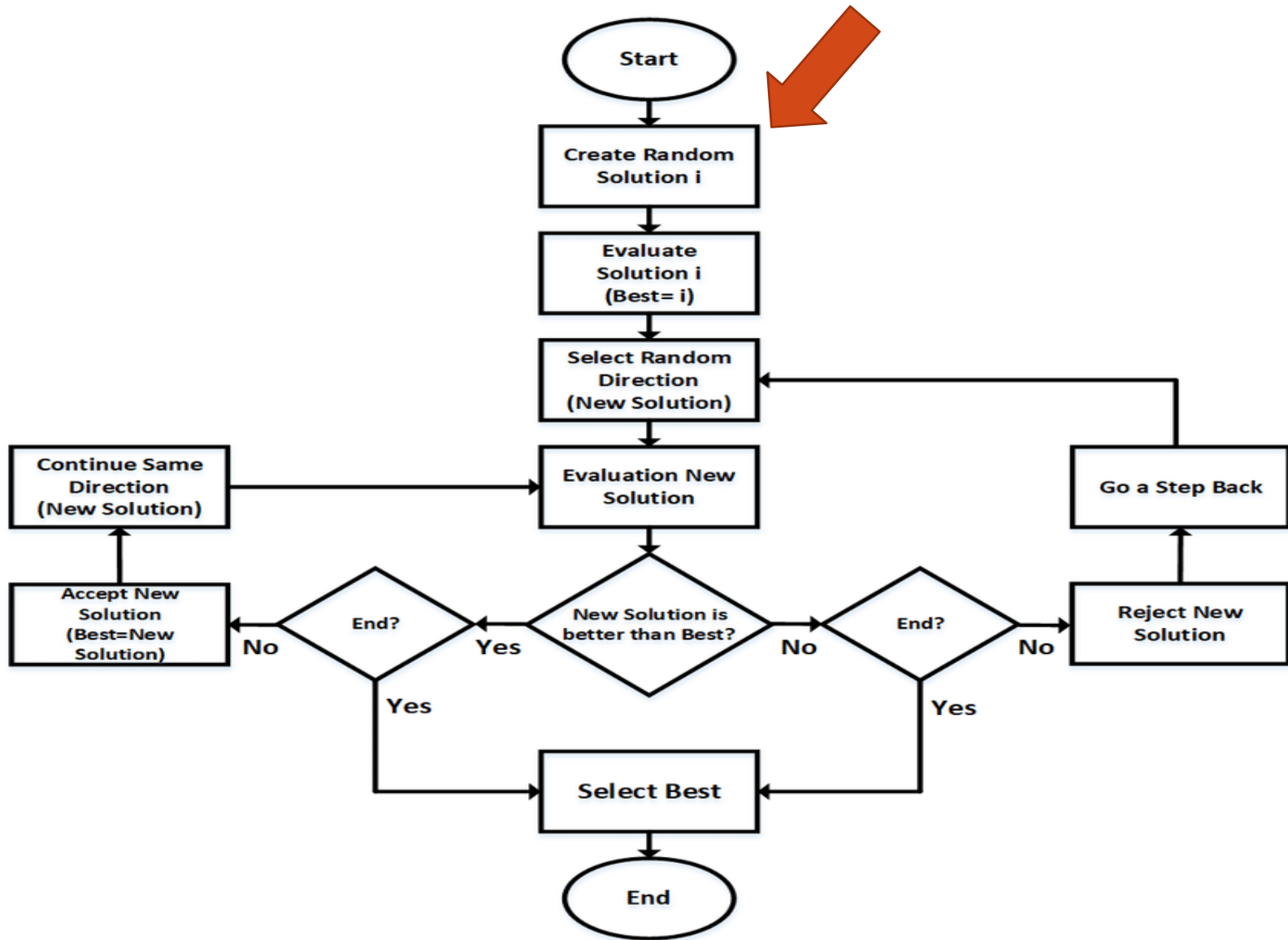
Stochastic optimization algorithm

- The use of randomness in the algorithms often means that the techniques are referred to as “**heuristic search**” as they use a rough rule-of-thumb procedure that may or may not work to find the optima instead of a precise procedure.
- Many stochastic algorithms are inspired by a **biological or natural process** and may be referred to as “**metaheuristics**” as a higher-order procedure providing the conditions for a specific search of the objective function. They are also referred to as “*black box*” optimization algorithms.

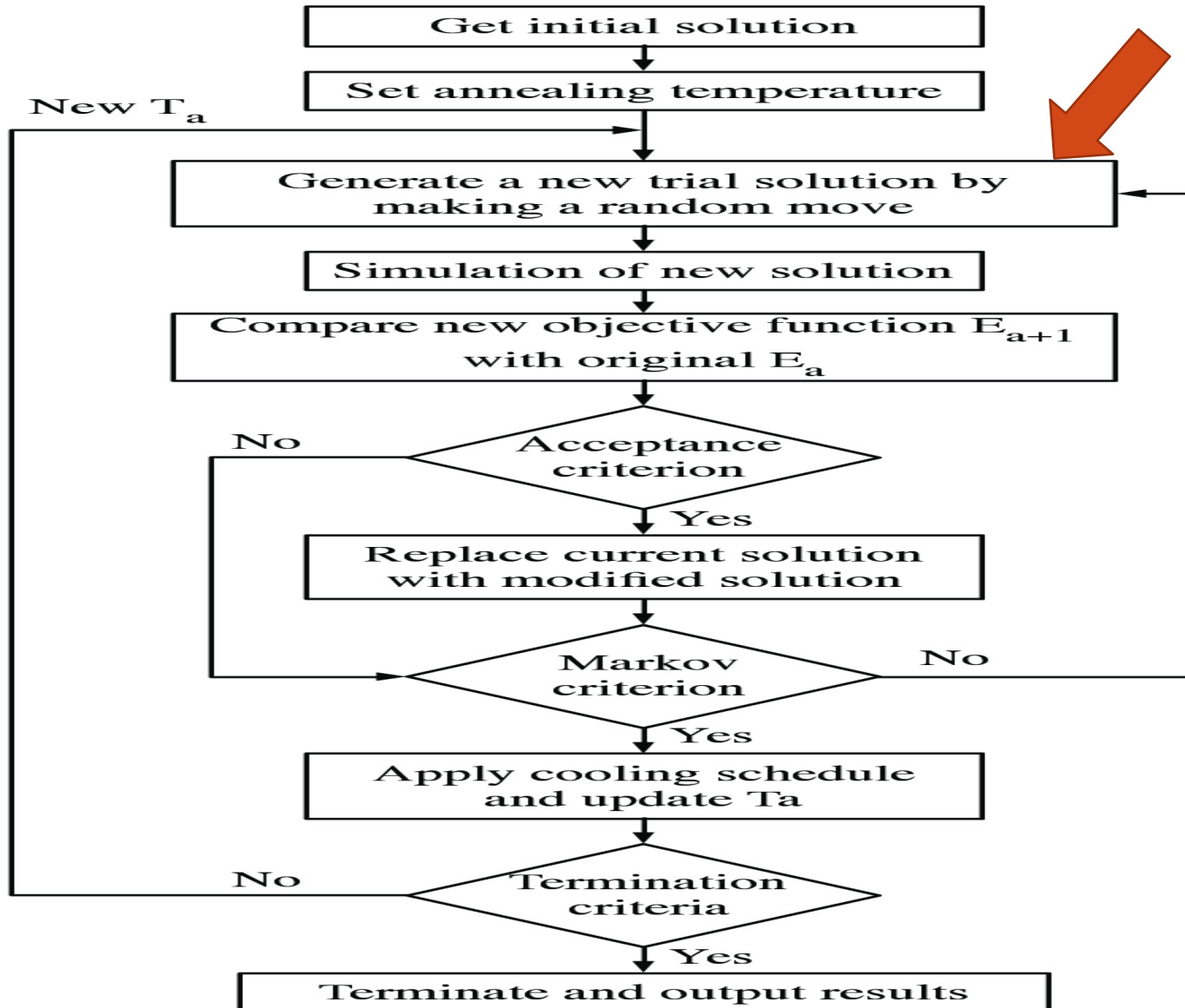
Examples of stochastic optimization algorithms:

- Iterated Local Search
- **Stochastic Hill Climbing**
- Stochastic Gradient Descent
- Tabu Search
- Greedy Randomized Adaptive Search Procedure

Hill Climbing Algorithm



Simulated annealing algorithm



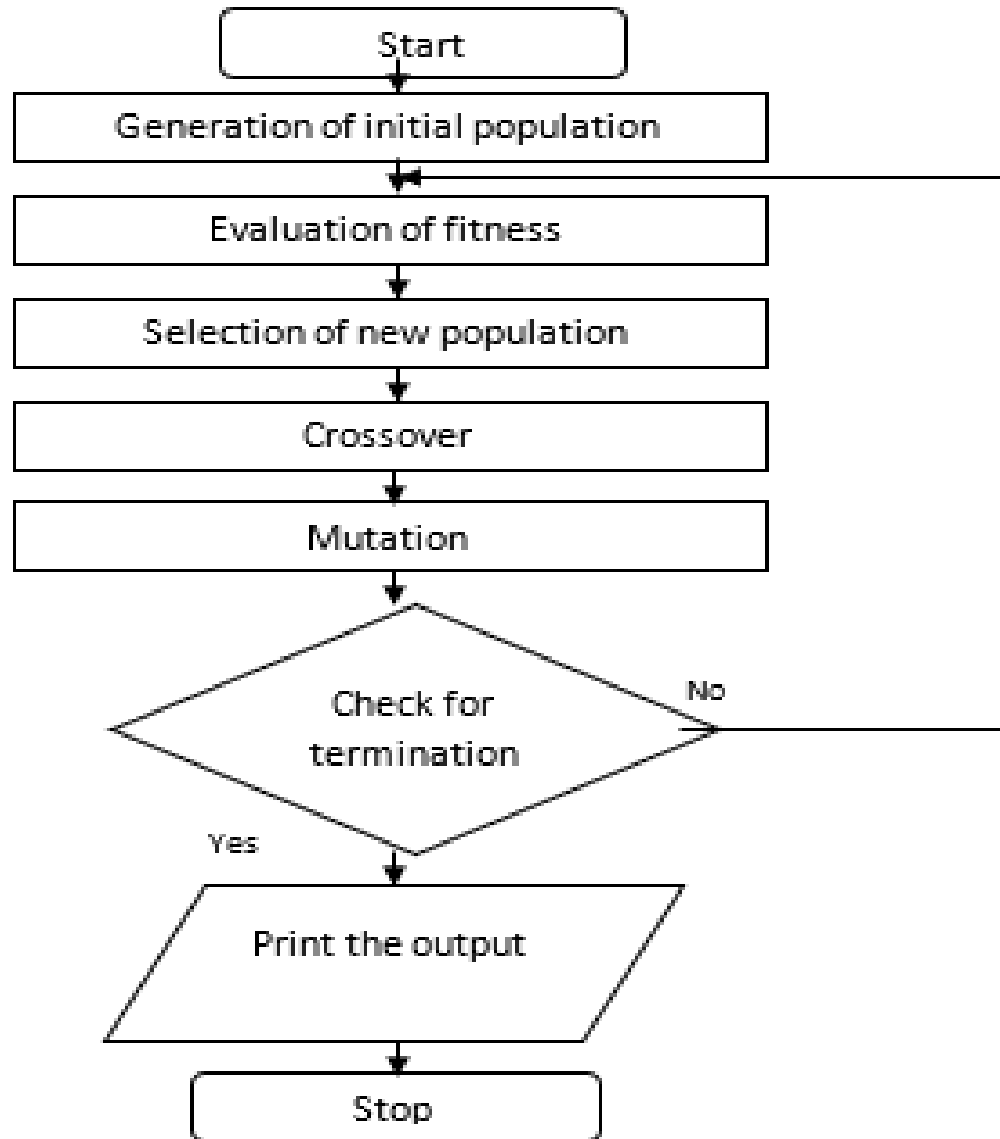
Stochastic optimization algorithm

- Stochastic optimization or stochastic search refers to an optimization task that involves randomness in some way, such as either from the objective function or in the optimization algorithm.
- **Genetic algorithms** and Particle swarm optimization (**PSO**) are good examples of stochastic algorithms.
- A **genetic algorithm** is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.
- In computer science and operations research, a **genetic algorithm** (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA)
- **Particle swarm optimization** (PSO) is one of the bio-inspired algorithms and it is a simple one to search for an optimal solution in the solution space.

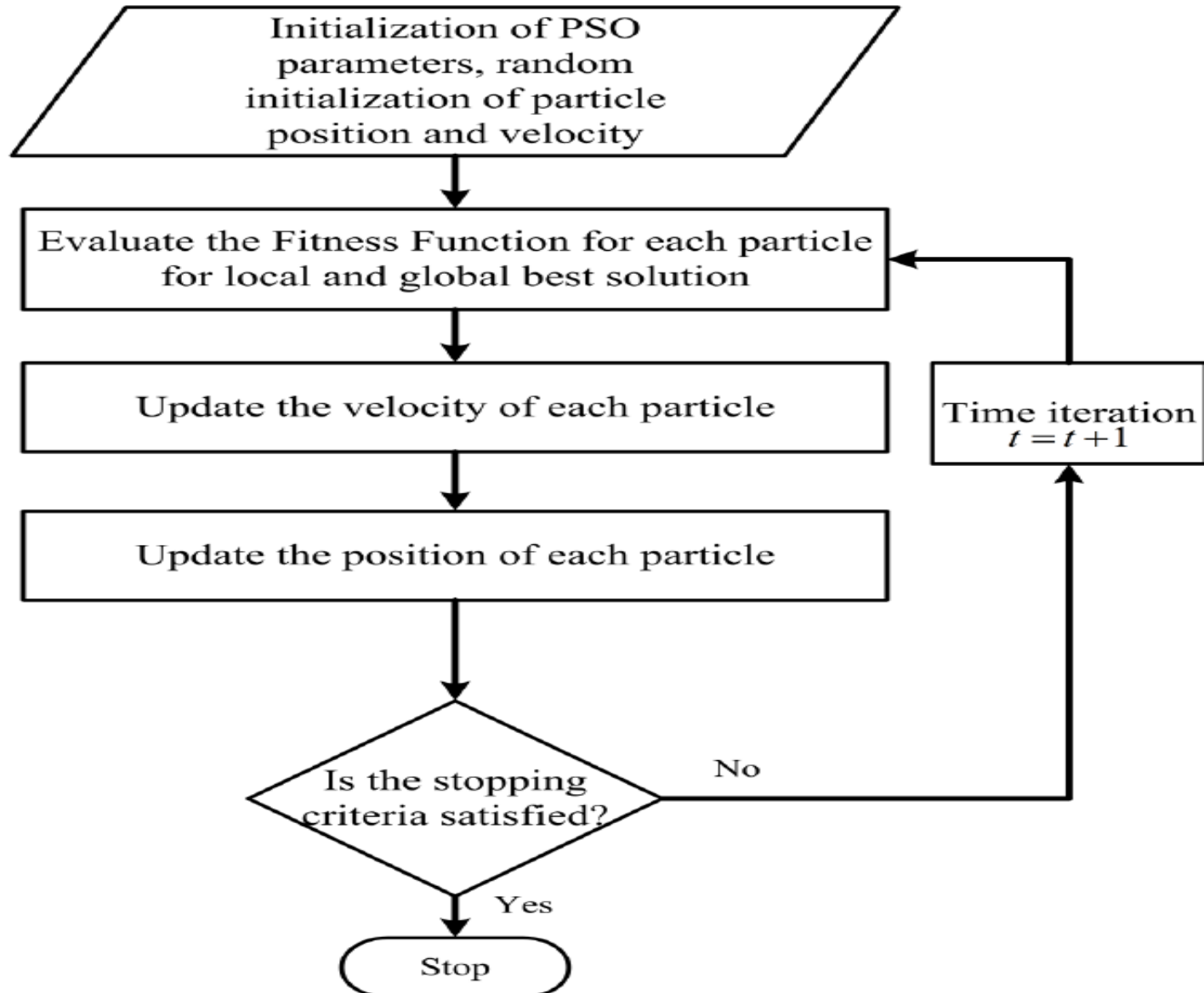
The steps of genetic algorithm

- Step 1: Generate random population of n chromosomes.
- Step 2: Evaluate the fitness $f_{(x)}$ of each chromosome x in the population.
- Step 3: Create a new population by repeating following steps until the new population is complete.
- Step 4: Select two parent chromosomes from a population according to the fitness.
- Step 5: With a crossover probability, crossover the parents to form a new offspring (Children). If no crossover was performed, offspring is an exact copy of parents.
- Step 6: With a mutation probability, mutate new offspring at each position in chromosome.
- Step 7: Place new offspring in a new population.
- Step 8: Use new generated population for a further run of algorithm.
- Step 9: If the end condition is satisfied, stop and return the best solution in current population and go to step 2.

Flowchart for genetic algorithm



Basic PSO algorithm



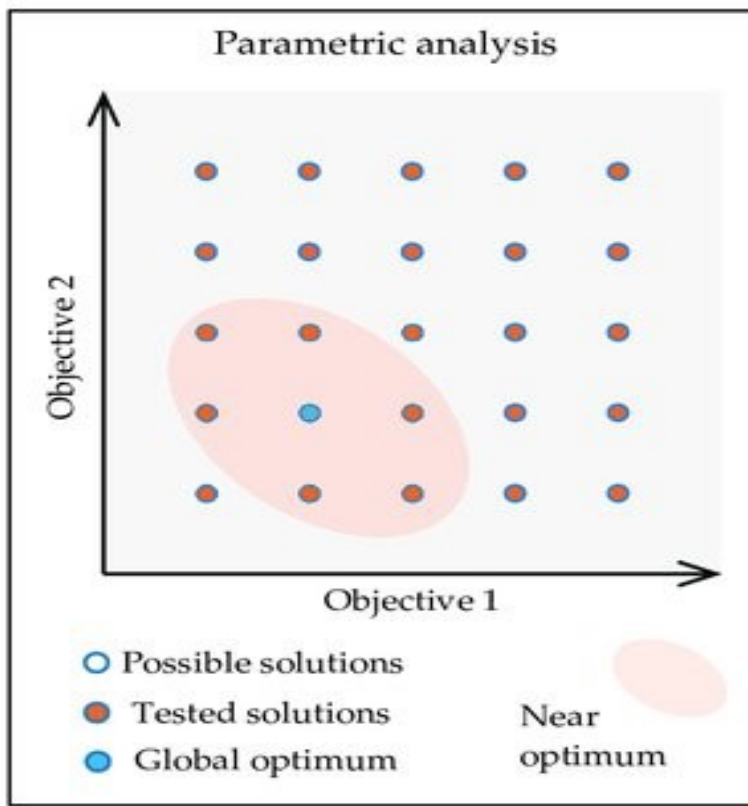
Metaheuristics

Examples of **stochastic optimization algorithms** that are inspired by biological or physical processes include:

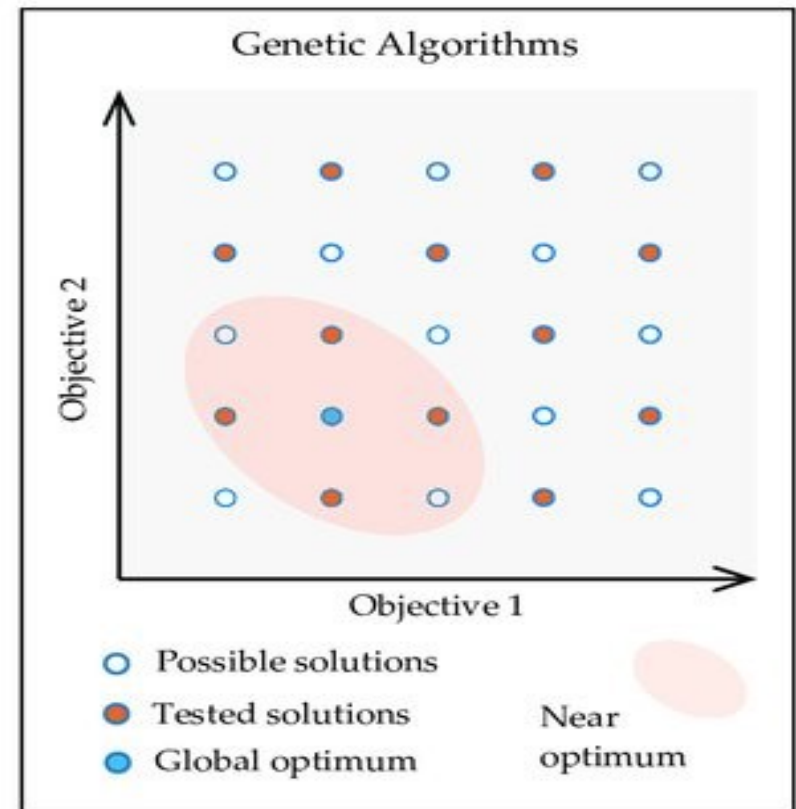
- Simulated Annealing
- Evolution Strategies
- Genetic Algorithm
- Differential Evolution
- Particle Swarm Optimization

Diagrammatic illustration of how genetic algorithms shorten the time-frame of a solution space search

In conventional parametric analysis (a) every possible solution is tested to identify the global optimum whereas when using genetic algorithms; and (b) a more efficient search is conducted utilizing evolutionary principles.



(a)

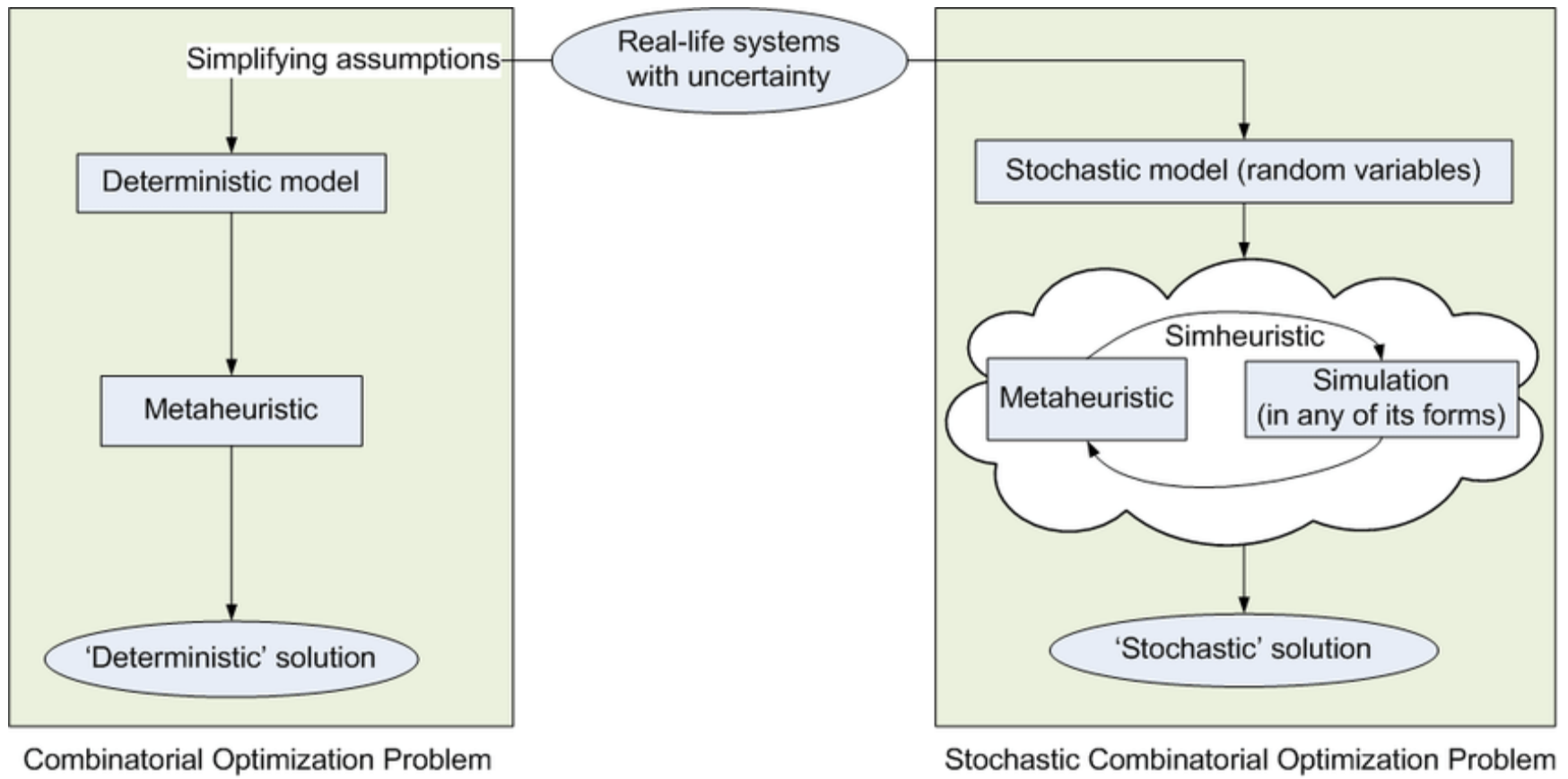


(b)

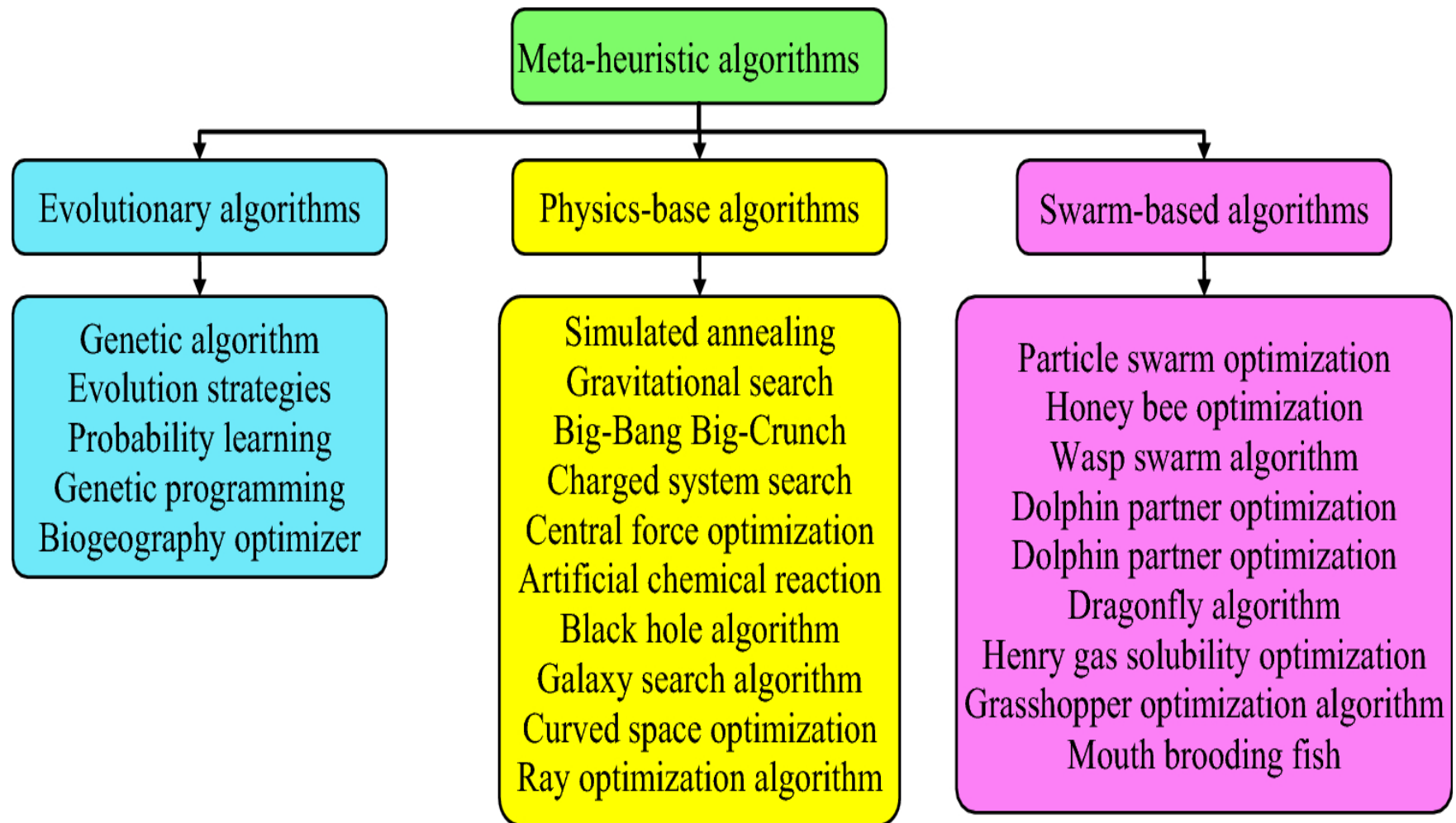
Multi-Restart

- **Multi-Restart:** An approach for improving the likelihood of locating the global optima via the repeated application of a stochastic optimization algorithm to an optimization problem.
- The repeated application of a stochastic optimization algorithm on an objective function is sometimes referred to as a multi-restart strategy and may be built in to the optimization algorithm itself or prescribed more generally as a procedure around the chosen stochastic optimization algorithm.

Combinatorial vs Stochastic optimization



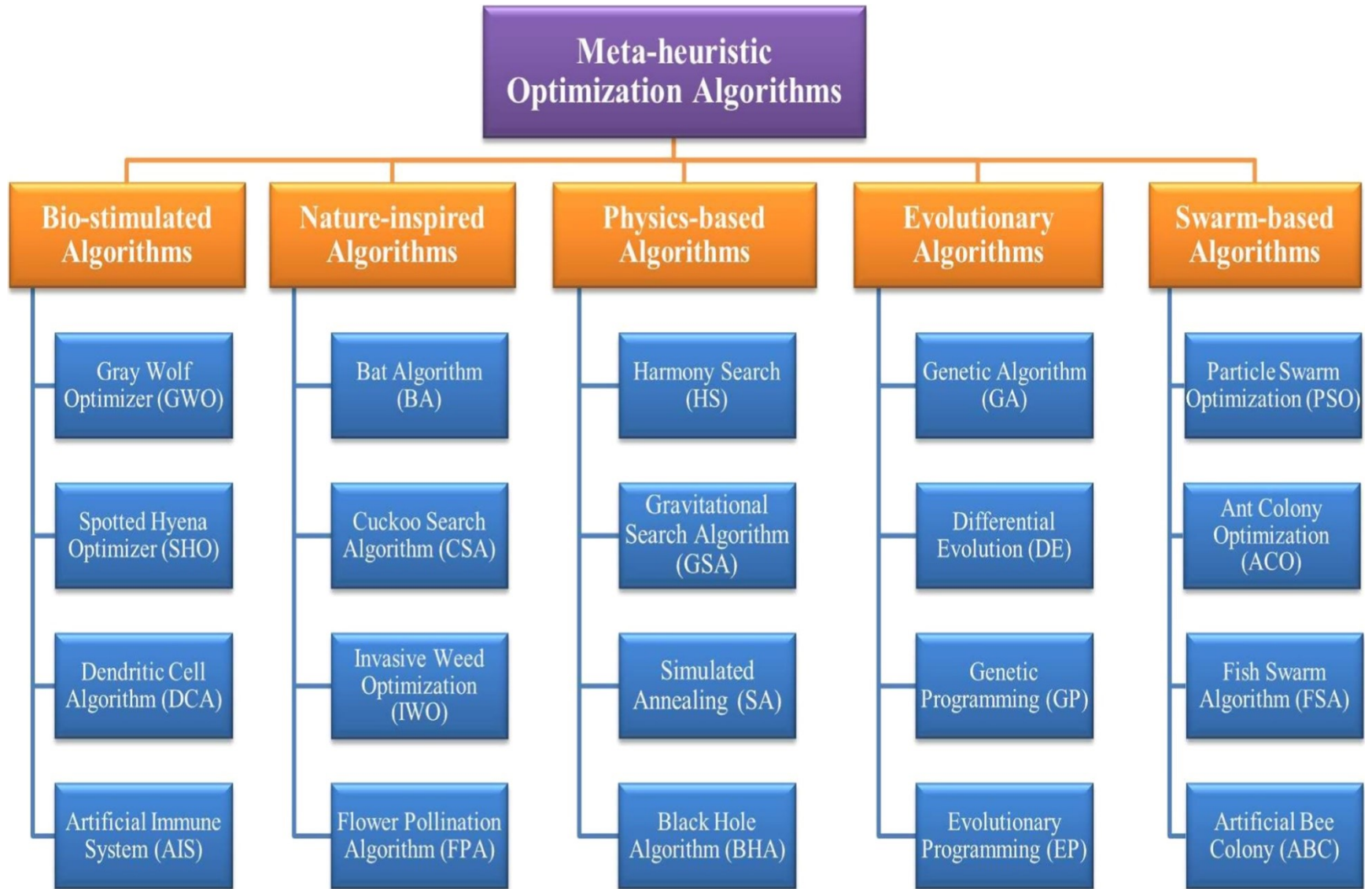
A taxonomy of metaheuristics



A taxonomy of metaheuristics



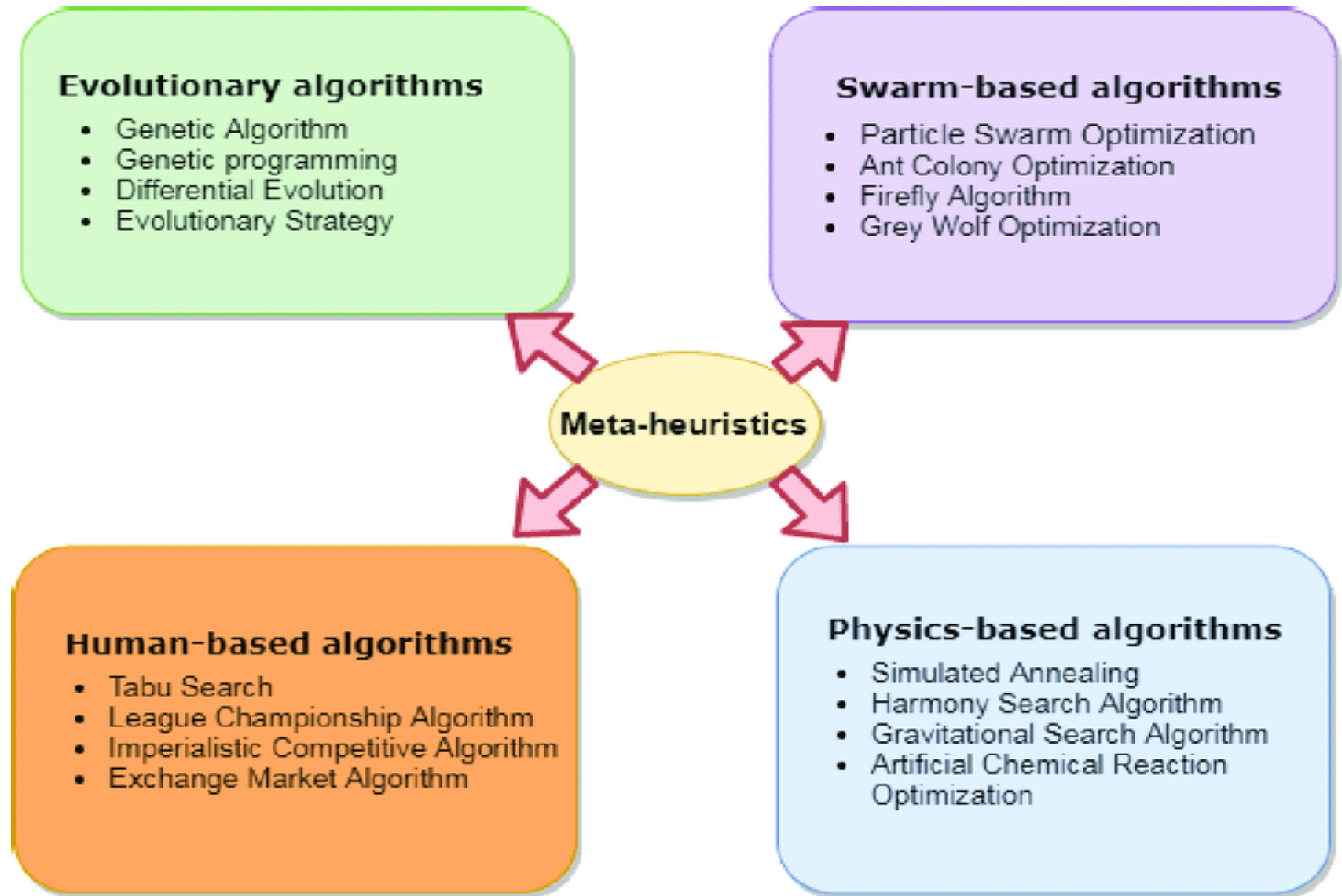
A taxonomy of metaheuristics



Metaheuristic optimization

- **Metaheuristic optimization** deals with **optimization** problems using **metaheuristic algorithms**.
- **Stochastic optimization** is the general class of algorithms and techniques which employ some degree of randomness to find optimal(or as optimal as possible) solutions to hard problems.
- **Metaheuristics** are the most general of these kinds of algorithms, and are applied to a very wide range of problems.
- A Metaheuristic optimization method can be considered as a more comprehensive intuitive method for the solution of optimization problems.
- All the metaheuristic methods are based on the use of **random numbers** (probabilistic approaches) in the various stages of the optimization process.
- Metaheuristics is a rather unfortunate term often used to describe a major subfield, indeed the primary subfield, of stochastic optimization.

Different categories of meta-heuristic algorithms



Structure of a metaheuristic computation algorithm

Most of the optimization methods have been designed to solve the problem of finding a global solution of a nonlinear optimization problem with box constraints in the following form:

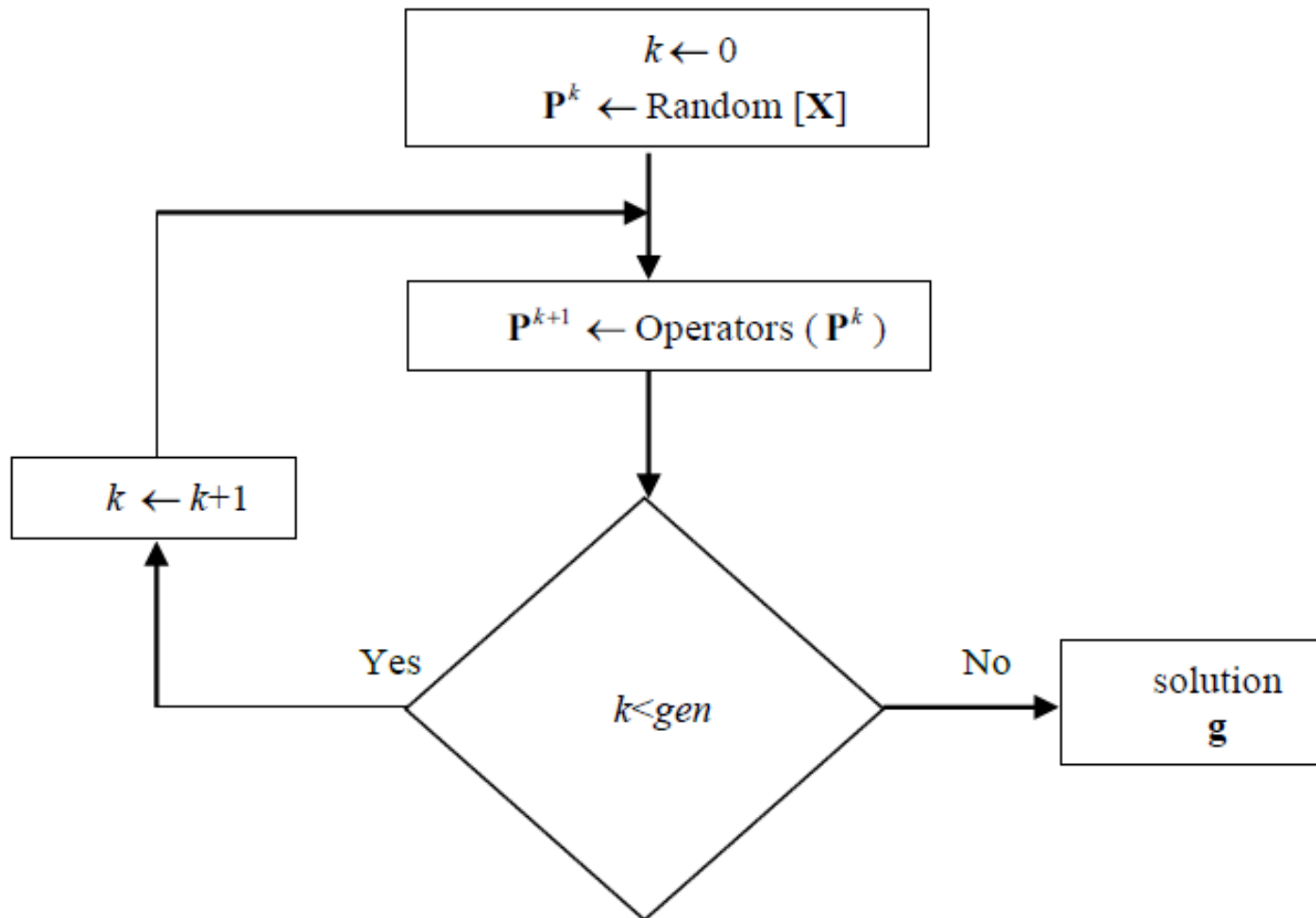
maximize	$f(\mathbf{x}), \quad \mathbf{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$	
subject to	$\mathbf{x} \in \mathbf{X}$	

where $f: \mathbb{R}^d \rightarrow \mathbb{R}$ is a nonlinear function whereas $\mathbf{X} = \{\mathbf{x} \in \mathbb{R}^d \mid l_i \leq x_i \leq u_i, i = 1, \dots, d\}$ is a bounded feasible search space, constrained by the lower (l_i) and upper (u_i) limits.

Structure of a metaheuristic computation algorithm

- The method: evolutionary computation. EC is a **computational intelligence technique inspired from natural evolution**. A standard EC algorithm includes:
 1. One or more populations of candidate solutions are considered.
 2. These populations change dynamically due to the production of new solutions.
 3. A fitness function reflects the ability of a solution to survive and reproduce.
 4. Several operators are employed in order to explore and exploit appropriately the space of solutions.
- The metaheuristic methodology suggests that, on average, candidate solutions improve their fitness over generations (i. e., their capability of solving the optimization problem).
- A simulation of the evolution process based on a set of candidate solutions whose fitness is properly correlated to the objective function to optimize will, on average, lead to an improvement of their fitness and thus steer the simulated population towards the global solution.

The basic cycle of a metaheuristic method



P^k is the population of k^{th} generation

Metaheuristics

- Metaheuristic methods obtain knowledge about the structure of an optimization problem by utilizing information obtained from the possible solutions (i.e., candidate solutions) evaluated in the past.
- This knowledge is used to construct new candidate solutions which are likely to have a better quality.
- Metaheuristic enables us to find optimized solutions to largely complicated problems by incorporating the following components:
 - ❑ Intensification and diversification
 - ❑ Exploitation and exploration

A taxonomy of metaheuristics

- Metaheuristic algorithms attempt to find the best (feasible) solution out of all possible solutions of an optimization problem.
- Metaheuristics operate on a representation or encoding of a solution, an object that can be stored in computer **memory** and can be conveniently **manipulated** by the different operators employed by the metaheuristic.
- Three fundamental classes of metaheuristics can be distinguished, based on the way in which solutions are manipulated.
 1. **Local search metaheuristics**
 2. **Constructive metaheuristics**
 3. **Population-based metaheuristics**

A taxonomy of metaheuristics

- **Local search metaheuristics** iteratively make small changes to a single solution.
- **Constructive metaheuristics** construct solutions from their constituting parts.
- **Population-based metaheuristics** iteratively combine solutions into new ones.
- However, these classes are not mutually exclusive and many metaheuristic algorithms combine ideas from different classes. Such methods are called **hybrid metaheuristics**.

1: Local search metaheuristics

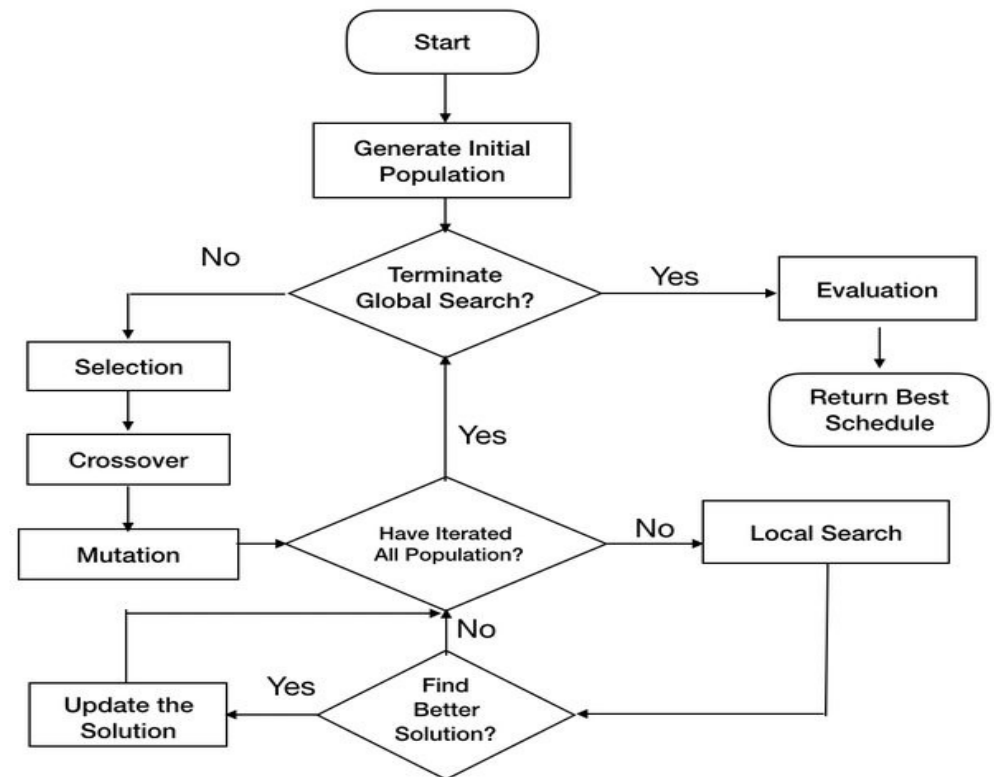
- **Local Search(LS)** is a term in applied mathematics and computer science defining a modification of local search or hill climbing methods for solving discrete optimization problems.

2: Constructive metaheuristics

- A **constructive heuristic** is a type of heuristic method which starts with an empty solution and repeatedly extends the current solution until a complete solution is obtained.
- It differs from local search heuristics which start with a complete solution and then try to improve the current solution further via local moves.

3: Population-based metaheuristics

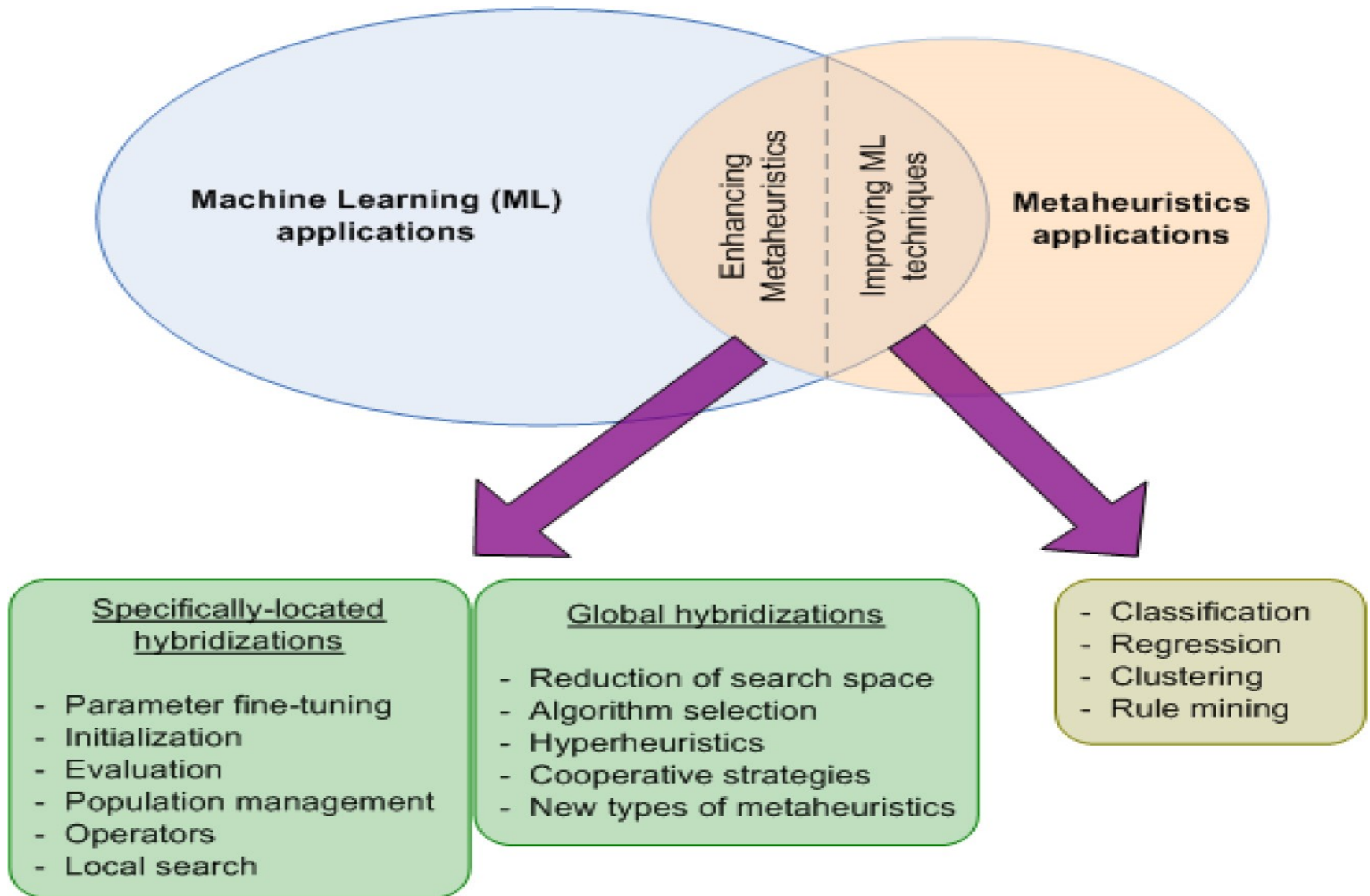
- Population-based approaches maintain and improve multiple candidate solutions, often using population characteristics to guide the search;
- Population based metaheuristics include **evolutionary computation, genetic algorithms, and particle swarm optimization.**



“Hybrid” metaheuristics

- The hybridization of metaheuristics with machine learning techniques is an emerging research field in the OR community.

Classification of works combining metaheuristics and machine learning



1:: Continuous optimization

- **Continuous optimization:** Metaheuristics are predominantly used for combinatorial optimization, but can be effectively adapted for continuous optimization, although this adaptation process is more involved for some metaheuristics than for others.
- The field of metaheuristics for continuous optimization has developed more or less independently from the field of combinatorial optimization, and a more complete reflection on the achievements in this field is out of the scope of this article

2::Multi-objective optimization

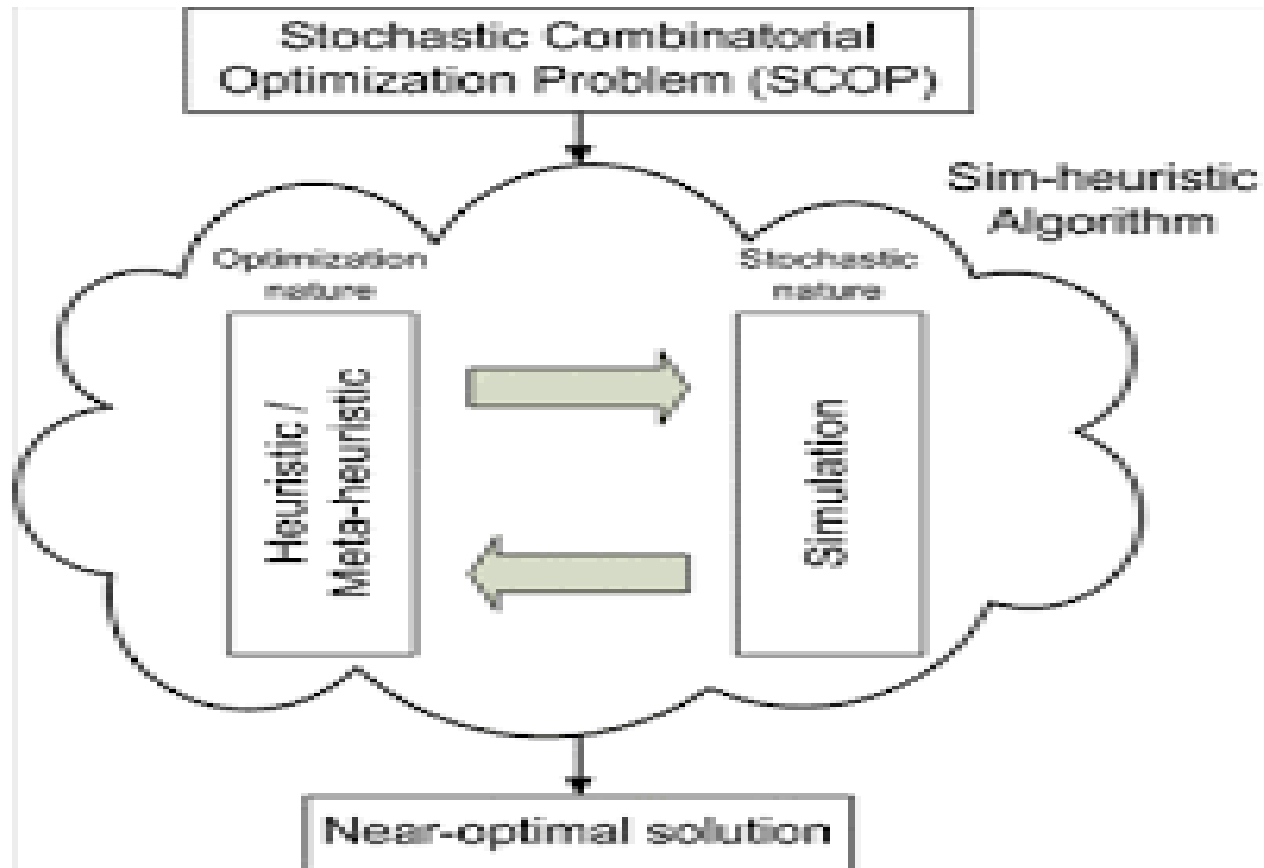
- Many optimization problems have multiple (conflicting) objectives, essentially rendering the concept of optimality meaningless since the best solution for one objective may not be the best for another. In multi-objective optimization the concept of dominance is therefore introduced. A solution is said to dominate another solution if its quality is at least as good on every objective and better on at least one. The set of all non-dominated solutions of an optimization problem is called the Pareto set and the projection of this set onto the objective function space is called the Pareto front.
- The aim of multi-objective metaheuristics is to approximate the Pareto front as closely as possible (Zitzler et al., 2004) and therefore generate a set of mutually non-dominated solutions called the Pareto set approximation.

•

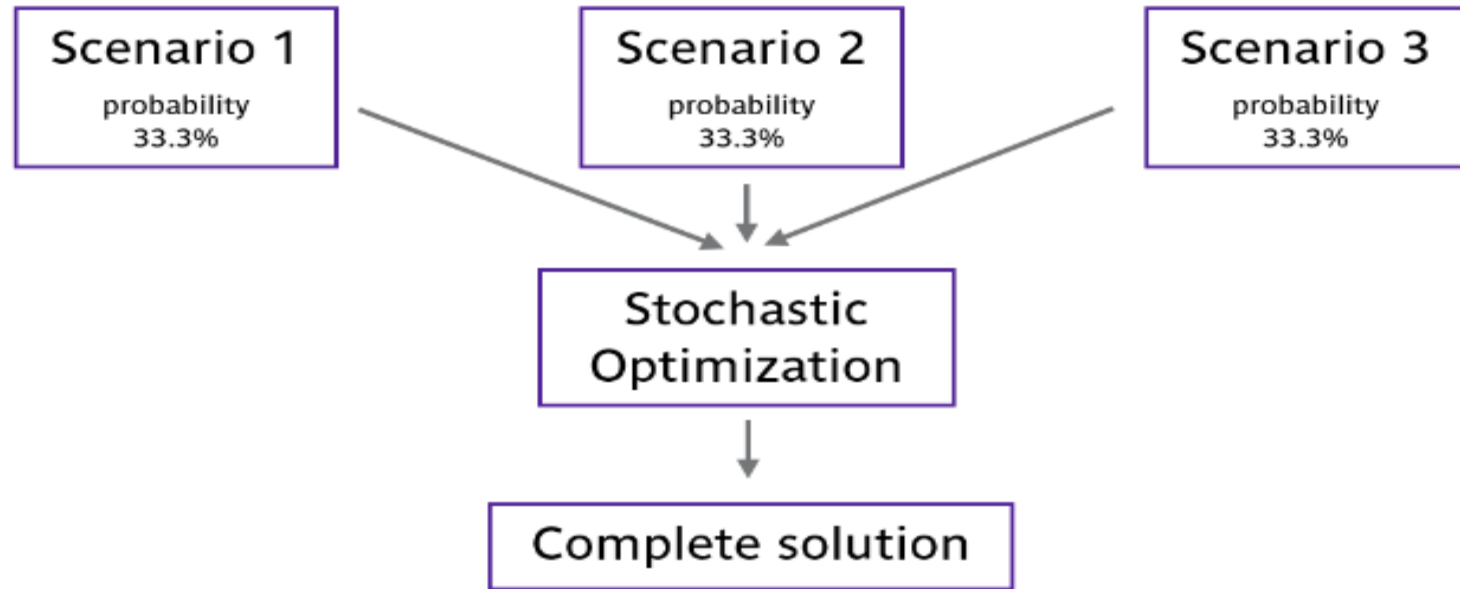
3:: Stochastic optimization

- Stochastic optimization refers to a collection of methods for minimizing or maximizing an objective function when randomness is present.
- Stochastic optimization is the process of maximizing or minimizing the value of a mathematical or statistical function when one or more of the input parameters is subject to randomness.
- Stochastic (combinatorial) optimization problems include uncertain, stochastic or dynamic information in their parameters.
- The objective function value and the violation of constraints of such problems are therefore random variables.
- Evaluating a solution's objective function value and/or its feasibility can be done either exactly (if a closed-form expression is available), by approximation or by Monte Carlo simulation.

3:: Stochastic optimization



A stochastic optimization procedure



- Stochastic optimization deals with optimization problems where the uncertainty is determined by probabilities.
- The optimization is performed for all scenarios combined. Therefore, the result is an overall solution which can handle every scenario under consideration and thus guard against uncertainty as best as possible.

3:: Stochastic optimization

- Stochastic optimization lends itself to real-life situations because many phenomena in the physical world involve uncertainty, imprecision or randomness.
- Stochastic processes are commonly involved in business analytics (BA), sales, service, manufacturing, finance and communications.
- Stochastic processes always involve probability, such as trying to predict the water level in a reservoir at a certain time based on random distribution of rainfall and water usage, or estimating the number of dropped connections in a communications network based on randomly variable traffic but constant available bandwidth.
- In contrast, deterministic processes never involve probability; outcomes occur (or fail to occur) based on predictable and exact input values.

3:: Stochastic optimization

- ❑ **Example:** A computer repair shop wants to order exactly the right number of spare parts of several different types every month to keep pace with customer demand.
- ❑ If the shop orders too many parts of any type from the wholesalers, money will be spent needlessly; if the shop does not order enough parts of any type, it will lose business when customers go elsewhere for service.
- ❑ Determining the ideal number of parts of each type to order involves stochastic optimization, because the number of customers who come in with component failures of various sorts cannot be precisely predicted.
- ❑ The objective is to maximize the function's output value (the shop's profit) in the face of numerous random input variables.

Metaheuristics software

- Frontline Systems' Risk Solver Platform and its derivatives, an extension of the Microsoft Excel solver, include a hybrid evolutionary solver.
- Tomlab/GENO is a package for static or dynamic, single- or multi-objective optimization based on a real-coded genetic algorithm.
- Both LINDO/LINGO and CPLEX include the relaxation induced neighborhood search (RINS) metaheuristic.
- The COIN-OR library has several (open source) metaheuristics software packages: METSlib, an object oriented metaheuristics optimization framework, and Open Tabu Search (OTS), a framework for constructing tabu search algorithms.
- Combinatorial optimization simulation packages: an optimization tool (Fu, 2002).
- Autostat, included in AutoMod and Simrunner, included in ProModel both use evolutionary algorithms.
- Solutions from a variety of companies in the simulation industry, as well as general management service and consulting firms like Rockwell Software, Dassault Systemes, Flextronics, Halliburton, HP, Planview and CACI, employ Opttek Systems, Inc. software OptQuest, which uses tabu search and scatter search.

Summary

- Metaheuristics provide the means to manage the trade-off between performance and quality of solutions. This implies a strong connection of metaheuristics with real-life applications, as they are driven by a pragmatic purpose.
- Metaheuristics is quite active nowadays, and we can expect more novel metaheuristics implementations, surely inspired by real-world phenomena and situations.

References

1. <http://www.scholarpedia.org/article/Metaheuristics>
2. <https://in.mathworks.com/help/gads/what-is-the-genetic-algorithm.html>
3. <https://www.intechopen.com/books/ant-colony-optimization-techniques-and-applications/an-ant-colony-optimization-algorithm-for-area-traffic-control>
4. <https://www.sciencedirect.com/science/article/pii/S2405656115000723>
5. https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms
6. <https://alexlowe.io/the-problem-space/>
7. <https://www.javatpoint.com/search-algorithms-in-ai>
- <https://www.investopedia.com/terms/h/heuristics.asp>
- <https://bjopm.emnuvens.com.br/bjopm/article/view/425/633>
- http://www.scholarpedia.org/article/Metaheuristic_Optimization#Optimization_Algorithms
- <https://www.investopedia.com/terms/h/heuristics.asp>
- Munien, Chanaleä and Ezugwu, Absalom E.. "Metaheuristic algorithms for one-dimensional bin-packing problems: A survey of recent advances and applications" *Journal of Intelligent Systems*, vol. 30, no. 1, 2021, pp. 636-663. <https://doi.org/10.1515/jisys-2020-0117>
- <https://techvidvan.com/tutorials/ai-heuristic-search/>
- <https://www.baeldung.com/cs/greedy-vs-heuristic-algorithm>
- <https://www.verywellmind.com/availability-heuristic-2794824>



Colourbox

Thank you for your attention

Henrichs Algorithm Design, Prof. Bishudatta
Sharma@NIT-Rourkela

© November 2021