Metaheuristics

- Meta- Greek word for upper level methods
- Heuristics – Greek word heuriskein – art of discovering new strategies to solve problems.
- Exact and Approximate methods
- Exact
  - Math programming LP, IP, NLP, DP
- Approximate
  - Heuristics
- Metaheuristics used for
  - Combinatorial Optimization problems – a general class of IP problems with discrete decision variables and finite solution space. Objective function and constraints could be non-linear too. Uses relaxation techniques to prune the search space.
  - Constraint Programming problems – used for timetabling and scheduling problems. Uses constraint propagation techniques that reduces the variable domain. Declaration of variables is a lot more compact

Need for Metaheuristics

- What if the objective function can only be simulated and there is no (or an inaccurate) mathematical model that connect the variables
  - Math and constraint programming need exact math formulations, therefore cannot be applied to the above
- Large solution space- Ex: TSP, 5 cities have 120 solutions (5!), 10 cities have 10! ~ 3.6 million, 75 cities have $2.5X10^{109}$ solutions
- Ex: Parallel machine job scheduling: process n jobs on m identical machines. There are m^n solutions
  - Complexity - time and space complexity
  - Time is the number of steps required to solve a problem of size n. We are looking for the worst-case asymptotic bound on the step count (not an exact count). Asymptotic behavior is the limiting behavior as n tends to a large number.

Complexity of Algorithms

- An algorithm has a complexity f(n)= O(g(n)) if there exist positive constants n_0 and c such that forall $n>n_0$, $f(n)<=c.g(n)$. In this case f(n) is upper bounded by g(n).
  - P class – polynomial time
  - If g(n) is a polynomial

    $g(n) = a_k.n^k + ...+ a_1.n + a_0$

    Then the corresponding algorithm has $O(n^k)$ complexity.

    Shortest path algorithms such as Dijkstra's with n nodes. Worst case scenario is that each of the n nodes have n neighbors. The algorithm needs no more than n^2 steps (upper bound) to find the solution. The complexity is $O(n^2)$ – a quadratic polynomial.

    Other examples are minimum spanning tree, max flow network etc.

    The solutions for the above problems are tractable

Complexity of Algorithms

- Exponential time $O(c^n)$ where is c is a positive constant >1

  Search Time of an Algorithm as a function of Problem size

  - Ex:

	Size	n= 10	n=20	n=30	n=40	n=50
Linear	O(n)	0.00001s	0.00002s	0.00003s	0.00004s	0.00005s
Quadratic	$O(n^2)$	0.0001s	0.0004s	0.0009s	0.0016s	0.0025s
5th order Poly	$O(n^5)$	0.1s	0.32s	24.3s	1.7 min	5.2 min
Expo base 2	$O(2^n)$	0.001s	1s	17.9min	12.7days	35.7years
Expo base 3	$O(3^n)$	0.059s	58 min	6.5 yrs	3855 centuries	$2X10^8$ centuries
Factorial	O(n!)	3.62 s	77146 years	$8.4 X 10^{16}$ centuries	~~~~too large~~~	

- NP class – non-deterministic polynomial time, the solutions for the problems are intractable

- NP- hard problems (a more generic name) need exponential time to find optimal solutions. Most real world optimization problems fit this category. Provably efficient algorithms do not exist. Metaheuristics can help to solve (near optimally) this class of problems.
- NP-complete problems are in NP class. A solution to NP-complete can be verified in polynomial time, however finding a solution is very difficult. These are also NP-hard and the hardest among NP hard problems. Typically decision making problems.

  May not be even decision problems such as subset sum problems.

  – Ex: TSP, scheduling problems such as flow-shop, job-shop scheduling, n jobs – m machine scheduling, quadratic assignment and location problems, grouping problems such as data clustering, graph partitioning, routing and set-covering, knapsack and so on.

Metaheuristics

- The idea: search the solution space directly. No math models, only a set of algorithmic steps, iterative method.
- Trajectory based methods (single solution based methods)
  - search process is characterized by a trajectory in the search space
  - It can be viewed as an evolution of a solution in (discrete) time of a dynamical system
    - Tabu Search, Simulated Annealing, Iterated Local search, variable neighborhood search, guided local search
- Population based methods
  - Every step of the search process has a population – a set of- solutions
  - It can be viewed as an evolution of a set of solutions in (discrete) time of a dynamical system
    - Genetic algorithms, swarm intelligence - ant colony optimization, bee colony optimization, scatter search
- Hybrid methods
- Parallel metaheuristics: parallel and distributed computing- independent and cooperative search

You will learn these techniques through several examples

History of Metaheuristics

- 1965: first Evolution Strategy
- 1975: first Genetic Algorithms
- 1983: Simulated Annealing
- 1986: Tabu Search
- 1991: Ant Colony Optimization
- 1997: Variable Neighborhood Search
- 2000+: parallel and distributed computing in metaheuristics

Metaheuristics

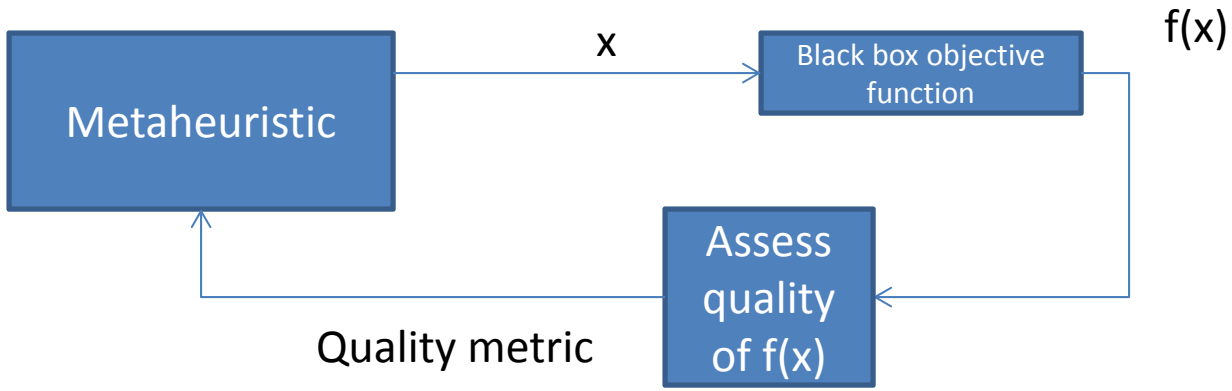
- The idea: search the solution space directly. No math models, only a set of algorithmic steps, iterative method. Find a feasible solution and improve it. A greedy solution may be a good starting point.
- Goal: Find the best solution for a given stopping criteria.
- Applied to combinatorial and constraint optimization problems
- Diversification and intensification of the search are the two strategies for search in Metaheuristics. One must strike a balance between them. Too much of either one will yield poor solutions. Remember that you have only a limited amount of time to search and you are also looking for a good quality solution. Quality vs Time tradeoff.
  - For applications such as design decisions focus on high quality solutions (take more time) Ex. high cost of investment, and for control/operational decisions where quick and frequent decisions are taken look for good enough solutions (in very limited time) Ex: scheduling
  - Trajectory based methods are heavy on intensification, while population based methods are heavy on diversification.

Classifications in Metaheuristics

- Nature inspired (swarm intelligence from biology) vs nonnature inspired (simulated annealing from physics)
- Memory usage (tabu search) vs memoryless methods (local search, simulated annealing SA)
- Deterministic (tabu, local search) vs stochastic (GA, SA) metaheuristics
  - Deterministic – same initial solution will lead to same final solution after several search steps
  - Stochastic – same initial solution will lead to different final solutions due to some random rules in the algorithm
- Population based (manipulates a whole population of solutions – exploration/diversification) vs single solution based search (manipulates a single solution – exploitation/intensification).
- Iterative vs greedy. Iterative – start with a complete solution(s) and transform at each iteration. Greedy- start with an empty solution and add decision variables till a complete solution is obtained.

When to use Metaheuristics

- If one can use exact methods then do not use Metaheuristics.
- P class problem with large number of solutions. P-time algorithms are known but too expensive to implement
- Dynamic optimization- real-time optimization- Metaheuristics can reduce search time and we are looking for good enough solutions.
- A difficult NP-hard problem - even a moderate size problem.
- Problems where objective function is a black box, i.e. often simulated and has no/inaccurate math formulation for the objective function



Evaluation of Results

- Best Metaheuristic approach- does not exist for any problem. No proof of optimality exists and you don't care for one either.
- A heuristic approach designed for problem A does not always apply to problem B. It is context-dependent.
- Questions: Is the metaheuristic algorithm well designed and does it behave well? No common agreement exist in the literature.
- What do we look for during implementation
  - Easy to apply for hard problems
  - Intensify: should be able to find a local optima
  - Diversify: should be able to widely explore the solution space
  - A greedy solution is a good starting point for an iterative search

Evaluation of Results

- Evaluation criteria include:
  - Ability to find good/optimal solutions. No way to prove optimality unless all solutions are exhaustively enumerated, which is infeasible.
  - Comparison with optimal solutions (if available). Test the heuristics on small problems before scaling it up.
  - Comparison with Lower/Upper bounds if already known (may be a target was already set)
  - Comparison with other metaheuristics methods
- Reliability and stability over several runs
  - Average gap between solution while approaching the (local) optimum
  - statistical tools (standard deviation of solutions, Confidence intervals, ANOVA)
- Reasonable computational time Usually: between a few seconds upto a few hours

Definitions

- Neighborhood of solution s is denoted a N(s).
- Local minima: solution s’ is local minima if forall s in N(s’), f(s’)<=f(s). Strictly local minima if f(s’)< f(s)

Main concepts to implement a Metaheuristic algorithm

- Representation of solutions
  - Vector of Binary values – 0/1 Knapsack, 0/1 IP problems
  - Vector of discrete values- Location , and assignment problems
  - Vector of continuous values on a real line – continuous, parameter optimization
  - Permutation – sequencing, scheduling, TSP
- Objective function
- Constraint handling
  - Reject strategies- keep only feasible solution and reject infeasible ones automatically.
  - Penalizing strategy - infeasible solutions are considered and a penalty function is added to the objective function for including a infeasible solution during the search. s_t, s_{t+1}, s_{t+2} are the search sequence where s_{t+2} and s_t are feasible but s_{t+1} is infeasible and $f(s_{t+2})$ is better than $f(s_t)$. The penalized obj function is $f'(s) = f(s) + \lambda\, c(s)$ where c(s) is the cost of infeasible s and λ is a weight.

Metaheuristics

- Next class
  - Single-solution metaheuristics