UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Combinatorial optimization

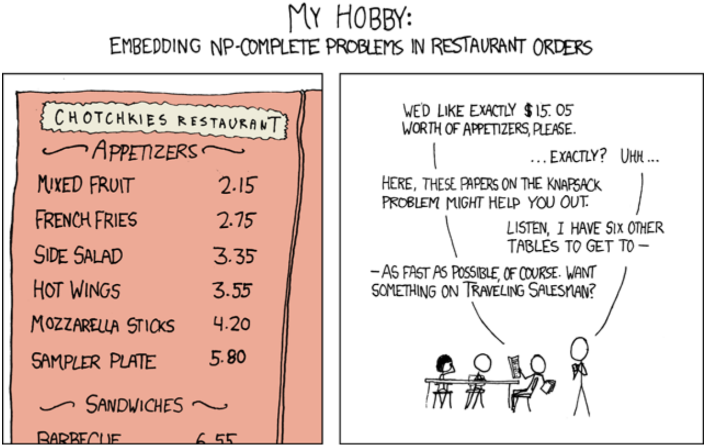
Jose M Sallan

Outline

Combinatorial optimization

The aim of combinatorial optimization is to find the object that optimizes (maximizes, minimizes) an **objective function** from **a finite set of objects**

Combinatorial optimization



Source: XKCD

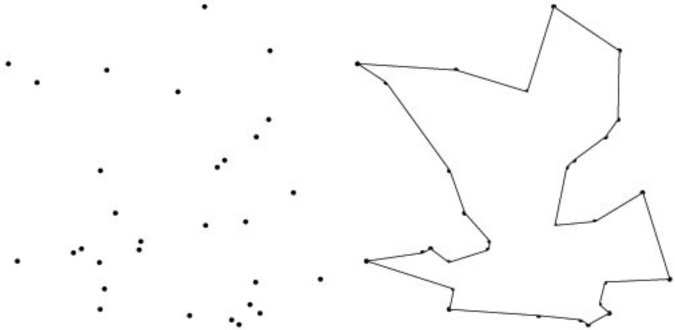
Knapsack problem

Knapsack problem (KP): given a set of objects of weight w_i and value u_i, select the objects to collect to pack in a container of maximum capability W maximizing the total value of the selected items.

Bin packing problem (BPP): a set of objects of volume v_i must be packed in bins of volume V minimizing the total number of bins used.

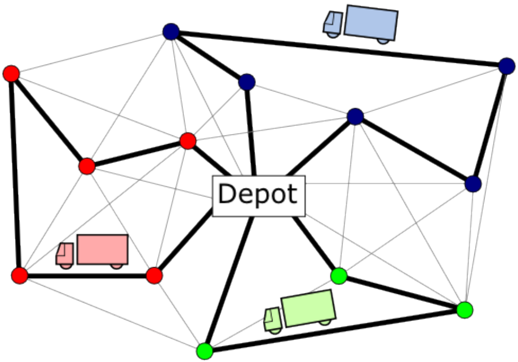
Traveling salesperson problem

Travelling salesman (salesperson) problem (TSP): given the distances between a collection of nodes, find the cycle that visits each node exactly once with the minimal total distance.



An example of TSP (source: WolframMathWorld)

Vehicle routing problem (VRP): given a depot and a set of nodes, define the optimal routes from the depot that satisfy a set of constraints (time, capacity, time windows...).



An illustration of the VRP (source: Universität Dortmund)

Two scheduling problems

Job shop problem: we have a set of $j = 1 : n$ jobs consisting in doing tasks in $i = 1 : m$ machines. Each job has a specific sequence of tasks along the machines, each task having a specific time t_{ij}. We must define the sequence of tasks for each machine to optimize a parameter representing process efficiency (e.g., makespan).

Flow shop problem: a specific case of job shop problem where all jobs are done using the machines in the same order. In **permutative flow shop**, each machine processes jobs in the same order.

Two scheduling problems

Job shop problem: we have a set of $j = 1 : n$ jobs consisting in doing tasks in $i = 1 : m$ machines. Each job has a specific sequence of tasks along the machines, each task having a specific time t_{ij}. We must define the sequence of tasks for each machine to optimize a parameter representing process efficiency (e.g., makespan).

Flow shop problem: a specific case of job shop problem where all jobs are done using the machines in the same order. In **permutative flow shop**, each machine processes jobs in the same order.

An instance of the flow shop problem

A permutative flow shop problem of $n = 4$ tasks and $m = 3$ machines

Table: Time of tasks in each machine

	T1	T2	T3	T4
M1	8	3	6	9
M2	4	5	3	2
M3	1	3	7	9

- How many solutions does this instance have?
- How long does it take to complete the tasks (makespan) for scheduling sequence T3, T1, T4, T2?

An instance of the flow shop problem

Table: Calculation of makespan for the sequence T3, T1, T4, T2

	T3	T1	T4	T2
M1	6	14	23	26
M2	9	18	25	31
M3	16	19	34	37

Permutative flow shop problem: Which permutation will yield the minimum makespan?

Solving combinatorial problems

As possible solutions of a CP are finite, a possible approach can be to **enumerate all solutions** an assess the objective function for each of them.

This may not viable in most cases. See why here...

So we need to think in ways to find:

- an **optimal** solution
- if the optimal solution is too costly to find, **a reasonably good solution** at a reasonable cost

Heuristics

A **heuristic** is any approach to problem solving, learning, or discovery that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals.

We use heuristics all the time in our life:

- Rule of thumb
- Stereotyping when judging others
- When looking for parking space
- using common sense

Metaheuristics

A **metaheuristic** is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms (Sörensen, 2015).

Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22(1), 3-18.

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

March (1991) posits that most adaptive processes are effective if they combine:

- *exploration:* search, variation, risk taking, experimentation
- *exploitation:* refinement, choice, efficiency

This principle has been used to define heuristics in:

- R&D management
- Theory formation
- Heuristics for combinatorial problems
- Parking space search

Algorithms

When applied to combinatorial problems, heuristics lead to definition of algorithms.

An **algorithm** is an unambiguous specification of how to solve a class of problems in a finite number of steps:

- Graph traversal algorithms: depth-first, breadth-first
- Sorting algorithms

An algorithm is usually expressed in **pseudocode**, and usually can be implemented in a **programming language**.

The **computational complexity** of an algorithm is the cost (time, steps) of applying the algorithm to a specific **instance**.

Metaheuristics for combinatorial problems

Constructive: try to build a solution with good elements

- TSP: smallest distances, biggest savings, nearest neighbor
- KP: relative utility ordering
- Flow shop: Johnson, Palmer, Companys...

Tree search: explore subsets of solutions arranged in a tree structure (branch and bound, branch and cut)
Local search: explore the neighbor solutions of a given solution (tabu search, simulated annealing)
Other: inspired in artificial intelligence and natural processes (evolutionary algorithms, swarm intelligence)

From metaheuristic to algorithm

How can we tackle a combinatorial problem:

- We can use a **metaheuristic** as a framework to define a **heuristic**
- The heuristic allows the definition of an **algorithm**, usually defining specific parameters
- The algorithm then is **coded** in a computer language
- Then, it can be **tested**, comparing its performance with other algorithms against a set of instances, finding a compromise between solution accuracy and execution time

Example: Taillard's scheduling instances

205053 Introduction to metaheuristics

An introduction to solve combinatorial problems using metaheuristics:

- Branch and bound
- Local search heuristics
- Evolutionary (genetic) algorithms

205054 Implementation and testing

An introduction to code and test algorithms:

- Introduction to coding
- Coding algorithms
- Testing algorithms through computational experiments