# Heuristic Algorithms

## Master's Degree in Computer Science/Mathematics

Roberto Cordone

DI - Università degli Studi di Milano

| | |
|---|---|
| Schedule: | Wednesday 13.30 - 16.30 in classroom Alfa |
| | Thursday 09.30 - 12.30 in classroom Alfa |
| Office hours: | on appointment |
| E-mail: | roberto.cordone@unimi.it |
| Web page: | https://homes.di.unimi.it/cordone/courses/2026-ae/2026-ae.html |
| Ariel site: | https://myariel.unimi.it/course/view.php?id=7439 |

# Effectiveness of a heuristic algorithm

A heuristic algorithm is useful if it is

1. **efficient**: it "costs" much less than an exact algorithm
2. **effective**: it "frequently" returns a solution "close to" an exact one

In particular, the **effectiveness** of heuristic algorithms is given by:

- closeness of the solution obtained to an optimal one
- frequency of hitting optimal or nearly optimal solutions

These features can be combined into a

- frequency distribution of solutions more or less close to the optimum

The effectiveness of a heuristic algorithm can be investigated with a

- **theoretical analysis** (*a priori*), proving that the algorithm finds always or with a given frequency solutions with a given guarantee of quality
- **experimental analysis** (*a posteriori*), measuring the performance of the algorithm on sampled benchmark instances to show that a guarantee of quality is respected in practice

We here discuss the theoretical analysis of effectiveness

# Indices of effectiveness

The effectiveness of a heuristic optimisation algorithm $A$ is measured by the difference between the heuristic value $f_A(I)$ and the optimum $f^*(I)$

- absolute difference:

$$\tilde{\delta}_A(I) = |f_A(I) - f^*(I)| \geq 0$$

  rarely used, and only when the objective is a pure number

- relative difference:

$$\delta_A(I) = \frac{|f_A(I) - f^*(I)|}{f^*(I)} \geq 0$$

  frequent in experimental analysis (*usually as a percent ratio*)

- approximation ratio:

$$\rho_A(I) = \max\left[\frac{f_A(I)}{f^*(I)}, \frac{f^*(I)}{f_A(I)}\right] \geq 1$$

  frequent in theoretical analysis: the first form is used for minimisation problems, the second one for maximisation problems

# Theoretical analysis (in the worst case)

To obtain a compact measure, independent from $I$, find the worst case
*(as for efficiency, that is complexity)*

The difference between $f_A(I)$ and $f^*(I)$ is in general unlimited,
but for some algorithms it is limited:

- absolute approximation:

$$\exists \tilde{\alpha}_A \in \mathbb{N} : \tilde{\delta}_A(I) \leq \tilde{\alpha}_A \text{ for each } I \in \mathcal{I}$$

    A (rare) example is Vizing's algorithm for *Edge Coloring* ($\tilde{\alpha}_A = 1$)

- relative approximation:

$$\exists \alpha_A \in \mathbb{R}^+ : \rho_A(I) \leq \alpha_A \text{ for each } I \in \mathcal{I}$$

Factor $\alpha_A$ ($\tilde{\alpha}_A$) is the relative (absolute) approximation guarantee

For other algorithms, the guarantee depends on the instance size

$$\rho_A(I) \leq \alpha_A(n) \text{ for each } I \in \mathcal{I}_n, n \in \mathbb{N}$$

Effectiveness can be independent from size (contrary to efficiency)

# How to achieve an approximation guarantee?

For a minimisation problem, the aim is to prove that

$$\exists \alpha_A \in \mathbb{R} : f_A(I) \leq \alpha_A f^*(I) \text{ for each } I \in \mathcal{I}$$

**1** find a way to build an underestimate $LB(I)$

$$LB(I) \leq f^*(I) \qquad I \in \mathcal{I}$$

**2** find a way to build an overestimate $UB(I)$,
related to $LB(I)$ by a coefficient $\alpha_A$

$$UB(I) = \alpha_A LB(I) \qquad I \in \mathcal{I}$$

**3** find an algorithm $A$ whose solution is not worse than $UB(I)$

$$f_A(I) \leq UB(I) \qquad I \in \mathcal{I}$$

Then $f_A(I) \leq UB(I) = \alpha_A LB(I) \leq \alpha_A f^*(I)$, for each $I \in \mathcal{I}$

$$f_A(I) \leq \alpha_A f^*(I) \text{ for each } I \in \mathcal{I}$$

# A 2-approximated algorithm for the *VCP*

Given a undirected graph $G = (V, E)$ find the minimum cardinality vertex subset such that each edge of graph is incident to it

A matching is a set of nonadjacent edges

Maximal matching is a matching such that any other edge of the graph is adjacent to one of its edges                    (*it cannot be enlarged*)
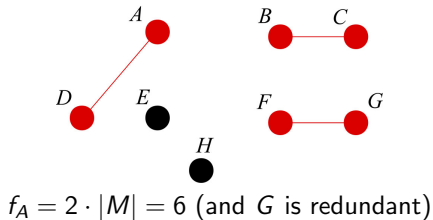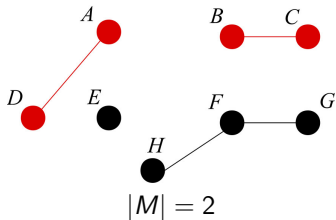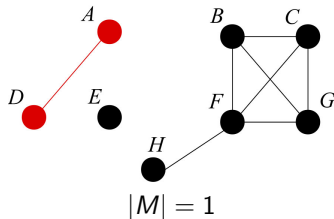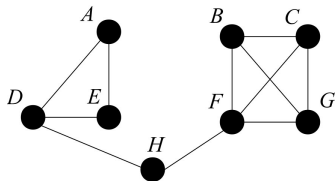
Matching algorithm:

**1** Build a maximal matching $M \subseteq E$ scanning the edges of $E$ and including in $M$ those not adjacent to $M$
                   (*now every edge of $E \setminus M$ is adjacent to an edge of $M$*)

**2** The set of extreme vertices of the matching edges is a *VCP* solution

$$x_A := \bigcup_{(u,v) \in M} \{u, v\}$$

and it can be improved removing the redundant vertices

$|M| = 1$

$|M| = 2$

$f_A = 2 \cdot |M| = 6$ (and $G$ is redundant)

The optimum is $f^* = 5$

The matching algorithm is 2-approximated

**1** The cardinality of matching $M$ is an underestimate $LB(I)$

- the cardinality of an optimal covering for any subset of edges $E' \subseteq E$ does not exceed that of an optimal covering for $E$

$$|x^*_{E'}| \leq |x^*_E|$$

(*it costs more to cover all edges than only the matching*)

- the optimal covering of a matching $M$ has cardinality $|M|$
  (*each edge of the matching requires exactly one different vertex*)

**2** Including both the extremes of each edge of the matching yields

- an overestimate (*it covers both the matching and the adjacent edges*)
- of value $UB(I) = 2LB(I)$ (*two different vertices for each edge*)

**3** The matching algorithm returns solutions of value $f_A(I) \leq UB(I)$
  (*possibly removing redundant vertices*)

This implies $f_A(I) \leq 2f^*(I)$ for each $I \in \mathcal{I}$, that is $\alpha_A = 2$

# . . . and the bound is tight!

Since $\alpha_A$ relates $UB(I)$ and $LB(I)$, $f_A(I)$ and $f^*(I)$ could be closer
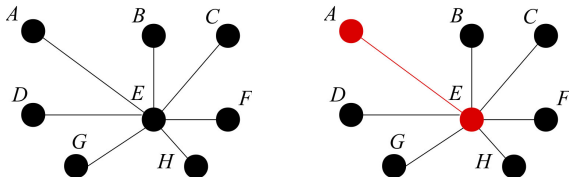
Actually, for many instances $\rho_A(I)$ is much better than $\alpha_A$

Are there instances $\bar{I}$ for which $f_A(\bar{I}) = \alpha_A f^*(\bar{I})$? How are they like?

The study of these instances is useful to

- evaluate whether they are rare or frequent
- introduce *ad hoc* modifications to improve the algorithm

In the literature the typical expression "*and the bound is tight*" introduces the description of instances exhibiting the worst case



*If all worst cases are patched, the approximation guarantee improves*

# The *TSP* under the triangle inequality

Consider the *TSP* with the additional (rather common) assumptions that
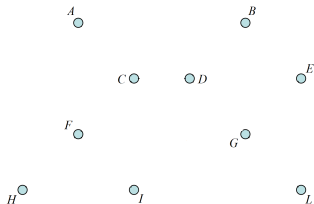
- graph $G = (N, A)$ is complete
- cost $c$ is nonnegative, symmetric and satisfies the triangle inequality

$$c_{ij} = c_{ji} \geq 0 \quad \forall i, j \in N \qquad \text{and} \qquad c_{ij} + c_{jk} \geq c_{ik} \quad \forall i, j, k \in N$$
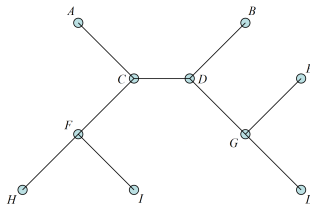
Double-tree algorithm

**1** Consider the complete undirected graph corresponding to $G$

**2** Build a minimum cost spanning tree $T^* = (N, X^*)$

**3** Make a pre-order visit of $T^*$ and build two lists of arcs:

    **a** $x'$ lists the arcs used both by the visit and the backtracking: this is a circuit visiting each node, possibly several times

    **b** $x$ lists the arcs linking the nodes in pre-order ending with the first: this is a circuit visiting each node exactly once
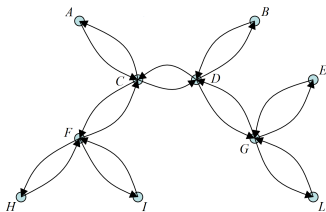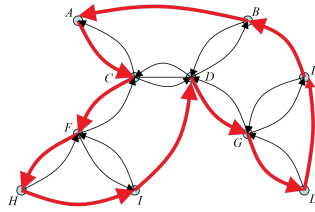
# Example



1) Complete graph $G$ (arcs omitted)

2. Minimum spanning tree $T^*$

3.a) $x' = (A, C, F, H, F, I, F, C, D, G, L, G, E, G, D, B, D, C, A)$

3.b) $x = (A, C, F, H, I, D, G, L, E, B, A)$

# Proof

The double-tree algorithm is 2-approximated

1. the cost of the minimum spanning tree is an underestimate $LB(I)$
   - deleting an arc from a Hamiltonian circuit yields a Hamiltonian path that is cheaper
   - a Hamiltonian path is a spanning tree (usually not of minimum cost)

2. the cost of circuit $x'$ is
   - an overestimate $UB(I)$      (*it is a nonminimum Hamiltonian circuit*)
   - equal to $2LB(I)$      (*two arcs correspond to each edge*)

3. the cost of circuit $x$ is $f_A(I) \le UB(I)$
        (*a single direct arc replacing a sequence decreases the cost*)

     This implies that $f_A(I) \le 2f^*(I)$ for each $I \in \mathcal{I}$, that is $\alpha_A = 2$

Notice: $x'$ is used in the approximation proof, but needs not be computed

# Inaproximability

For an inapproximable problem, approximated algorithms would be exact

Consider this family of *TSP* instances on complete graphs:

- $c_{ij} = 0$ for $(i,j) \in A_0$
- $c_{ij} = 1$ for $(i,j) \in (N \times N) \setminus A_0$  (*the triangle inequality is violated!*)

The optimum of any such instance $\bar{I}$ is:

$$\begin{cases} f^*\left(\bar{I}\right) = 0 \text{ if } A_0 \text{ contains a Hamiltonian circuit} \\ f^*\left(\bar{I}\right) \geq 1 \text{ otherwise} \end{cases}$$

(*in the latter case, the optimal solution contains at least an arc $\notin A_0$*)

Assume that a polynomial algorithm $A$ provide a guarantee $\alpha_A$

$$f^*\left(I\right) \leq f_A\left(I\right) \leq \alpha_A f^*\left(I\right) \ \forall I \in \mathcal{I}$$

Then $f^*\left(\bar{I}\right) = 0 \Leftrightarrow f_A\left(\bar{I}\right) = 0$

Whenever the subgraph $G\left(N, A_0\right)$ has a Hamiltonian circuit, $A$ finds it, solving an $\mathcal{NP}$-complete problem in polynomial time ($\mathcal{P} = \mathcal{NP}$)

(*unlikely*)

For hard problems

- exact algorithms provide the best approximation guarantee ($\alpha_A = 1$), but require exponential time $T_A$
- approximated algorithms provide a worse guarantee ($\alpha_A > 1$), but could require polynomial time $T_A$

Some problems admit a family of algorithms providing a whole range of compromises between efficiency ed effectiveness

- better and better approximation guarantees: $\alpha_{A_1} > \ldots > \alpha_{A_r}$
- worse and worse computational complexities: $T_{A_1} < \ldots < T_{A_r}$

Approximation scheme is a parametric algorithm $A_\alpha$ allowing to choose $\alpha$

(Example: the *KP*)

# Beyond the worst case

As usual, the worst-case approach is rough:
some algorithms often have a good performance, though sometimes bad

The alternative approaches are similar to the ones used for complexity

- parametrisation: prove an approximation guarantee that depends on other parameters of the instances besides the size $n$
- average-case: assume a probability distribution on the instances and evaluate the expected value of the approximation factor
  (*the algorithm could have a bad performance only on rare instances*)

but there is at least another approach

- randomisation: the operations of the algorithm depend not only on the instance, but also on pseudorandom numbers, so that the solution becomes a random variable which can be investigated
  (*the time complexity could also be random, but usually is not*)

# Randomised approximation algorithms

For a randomised algorithm $A$, $f_A(I, \omega)$ and $\rho_A(I, \omega)$ are random variables depending on the pseudorandom number seed $\omega$

A randomised approximation algorithm has an approximation ratio whose expected value is limited by a constant

$$E[\rho_A(I, \omega)] \leq \alpha_A \text{ for each } I \in \mathcal{I}$$

*Max-SAT* problem: given a CNF, find a truth assignment to the logical variables that satisfy a maximum weight subset of formulae

Purely random algorithm:

Assign to each variable $x_j$ $(j = 1, \ldots, n)$
- value *False* with probability $1/2$
- value *True* with probability $1/2$

*What is the expected value of the solution?*

# Randomised approximation for the *MAX-SAT*

The objective $f(x) = f_A(I, \omega)$ is the total weight of the satisfied clauses and its expected value is

$$E[f_A(I, \omega)] = E\left[\sum_{i \in \mathcal{C}} \delta_i(x(\omega)) w_i\right] = \sum_{i \in \mathcal{C}} (w_i \cdot Pr[\delta_i(x(\omega)) = 1])$$

where $\delta_i(x(\omega)) = 1$ if solution $x(\omega)$ satisfies clause $i$, 0 otherwise

Let $k_i$ be the number of literals of formula $i \in \mathcal{C}$ and $k_{\min} = \min_{i \in \mathcal{C}} k_i$

$$Pr[\delta_i(x(\omega)) = 1] = 1 - \left(\frac{1}{2}\right)^{k_i} \geq 1 - \left(\frac{1}{2}\right)^{k_{\min}} \quad \text{for each } i \in \mathcal{C}$$

$$\Rightarrow E[f_A(I, \omega)] \geq \sum_{i \in \mathcal{C}} w_i \cdot \left[1 - \left(\frac{1}{2}\right)^{k_{\min}}\right] = \left[1 - \left(\frac{1}{2}\right)^{k_{\min}}\right] \sum_{i \in \mathcal{C}} w_i$$

and since $\sum_{i \in \mathcal{C}} w_i \geq f^*(I)$ for each $I \in \mathcal{I}$ one obtains

$$\frac{E[f_A(I, \omega)]}{f^*(I)} \geq \left[1 - \left(\frac{1}{2}\right)^{k_{\min}}\right] \geq \frac{1}{2}$$