Heuristic Summary

Massimo Perego

Algorithm Analysis

Run Time Distribution Diagram RTD: It shows the probability that the execution time is below a certain time t (axis: runtime-P of solving). To draw it: how many of the instances are solved in that amount of time or less?

Scaling Diagram: Describes the dependence of the time on the size (axis: instance size-execution time). Logarithmic scale for the time shows that the algorithm is exponential, both axis logarithmic can show that the algorithm is polynomial. To find the coefficient: choose two points and

$$\left(\frac{s_1}{s_2}\right)^{\alpha} \approx \frac{T_1}{T_2}$$

find α, with s_1, s_2 and T_1, T_2 instance sizes and times respectively.

Solution Quality Diagram SQD: It plots the probability that the relative difference δ is smaller than a value α, for each possible value of α (axis: relative solution quality-cumulative frequency, i.e., how many times do we get that solution quality or better). To draw it: sort the relative differences by non-decreasing values and determine how many of the instances are under each value.

Boxplot diagram: Show median ($n/2$th element), surrounded by the box formed by the lower and upper quartiles ($n/4$th element and $3n/4$th element, respectively). The whiskers go out to the end of the range in both directions.

Dominance: The dominance can be:

- **Strict:** a boxplot is fully below the other;
- **Probabilistic:** each quartile is not above the corresponding one of the other algorithm.

Wilcoxon's Test: It's a test, focused on effectiveness, used to determine if the empirical difference among two algorithms is significant. Steps:

1. Compute the difference of the algorithms results, instance by instance
2. Sort them by increasing absolute values and assign a rank (number) to each one
3. Give a sign to each rank, positive if the difference was positive, negative otherwise
4. Sum all positive ranks (W^+), as well as the negative (W^-)
5. Calculate the probability that $|W^+ - W^-|$ is equal or larger than the observed value (not required during exams)

If the probability is low enough the difference is significant and if $W^+ > W^-$ then the first algorithm is better than the second, or vice versa.

Constructive Heuristics

Traveling Salesman Problem TSP: Types of Heuristic:

- **Cheapest Insertion:** adds a node and removes an arc from the circuit, such that the total change is minimum;
- **Nearest Insertion:** look for the node closest to the circuit and find the best way to include it;
- **Farthest Insertion:** search for the node farthest from the circuit and connect it in the best way possible.

Constructive Metaheuristics

Adaptive Research Technique ART: Set a number of iterations ℓ for a basic constructive heuristic, at each iteration forbid elements inside the solution with probability π for a number of iteration L. When wanting to include an element, check that the current iteration is distant enough from the last ban of said element, i.e., if $t > T_i + L$ where t is the current iteration and T is the vector containing the tabu attribute for each element.

Greedy randomized Adaptive Search Procedure GRASP: Also called semi-greedy, instead of always choosing the best at each step, sometimes make a random step. How can we determine which step?

- **Uniform probability:** each possibility has the same probability;
- **HBSS:** sort the possibilities by non-increasing values of φ, assign a probability according to the position in the order;
- **Restricted Candidate List:** sort the possibilities, choose from a number of choices among the best. We can define the RCL through
  - **Cardinality:** take the μ best elements
  - **Value:** include all the elements above the threshold

$$\varphi_\mu = (1-\mu)\varphi_{\max} + \mu\varphi_{\min}$$

Ant System: Similar to semi-greedy, but all choices are feasible and the probability of choosing an element is a function which depends on the selection criteria (φ, which is turned into visibility η) and some auxiliary information (trail τ), updated after each iteration. To update we have the oblivion parameter ρ and the conversion coefficient Q:

$$\tau_i = \begin{cases} (1-\rho)\tau_i & \text{for } i \notin x \\ (1-\rho)\tau_i + \rho \cdot 2Q \cdot f(x) & \text{for } i \in x \end{cases}$$

where $f(x)$ is the value of the final solution. We multiply all trails by $(1-\rho)$ and then add $\rho Q f(x)$ to the elements in the current solution, since they're "promising".