Swarm Intelligence

Other ACO Algorithms and ACOTSP

Leonardo Bezerra and Leslie Perez Caceres

IRIDIA – Université Libre de Bruxelles (ULB)
Bruxelles, Belgium
lperez@iridia.ulb.ac.be
leonardo@iridia.ulb.ac.be

What will be the project about?

Design, ***Implementation*** *and* ***evaluation*** *of*

ACO algorithms for solving an optimization problem.

- Could be like this:
  1. Research the problem
  2. Design the algorithms
  3. Analyse the parameters
  4. Compare them and choose the best
  5. Compare with the state of the art
  6. Interesting conclusions and ideas!

Facing the Problem

- Each problem has its own characteristics:
  - Use methods/components already proposed (don't reinvent the wheel)
  - Avoid things that wont work!
- Check the **state of the art** looking for:
  - Definition
    - Mathematical model
    - Variants
    - Instances
  - State of the art
    - Common representations
    - Heuristics
    - Best algorithms
      - complete techniques
      - constructive algorithms
      - local search algorithms ...
    - The general research state

Design the algorithm

- Design the components of your algorithm using what you know from the state of the art.
  - Structural decisions: what information will be stored and how
    - Representation: How the solutions will be stored?
      - Memory use
      - Evaluation function calculation
      - Operations over the solution
    - Other components
      - Ex. Heuristic information
  - Algorithmic decisions: how things will be performed
    - Operators and components
      - Ex. Pheromone update, transition rule
    - Constraint handling
      - Discard
      - Penalization
      - Representation
    - Evaluation function

Small example
Knapsack problem

A thief is choosing items to steal from a store. Since he has to leave the country intermediately he needs to fill his bag according to the airline standards. Choose a set of items to be carried in the bag that maximizes the income of the thief.

- In a formal way:
  - Given a set of ***n*** items, each item ***j*** having an integer profit $\boldsymbol{p_j}$ and an integer weight $\boldsymbol{w_j}$. Choose a subset of the items such that their overall profit is maximized, while the overall weight does not exceed a given capacity ***c***.

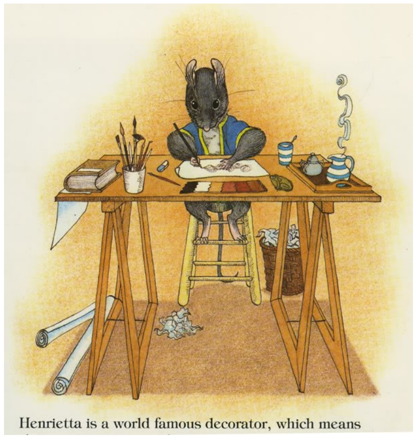
$$\begin{aligned} &\textit{maximize} && \sum_{j=1}^{n} p_j x_j \\ &\textit{subject}\ \text{to} && \sum_{j=1}^{n} w_j x_j \leq c \\ & && x_j \in \{0,1\},\ j=1,\dots,n \end{aligned}$$

Small example
Knapsack problem

- Structural decisions
  - Solution:
    - List of items that are selected T={1,2,6}
    - Array indicating if an item is selected or not T={1,1,0,0,0,1}
  - Heuristic/pheronome information:
    - Variable for each item (if we select or not)
    - Matrix for the selection of items i and j together.
- Algorithmic decisions
  - Heuristic information
    - Proportional to the value of the item $\eta_i = 1/v_i$, $\eta_i = v_i / sum(v_i)$
    - Proportional to its weight $\eta_i = 1/w_i$
    - Using weight and value
  - Pheromone update rule → AS, Elitist, Best worst...
  - Transition rule

Small example
Knapsack problem

- Algorithm design decisions
  - Constraints
    - Is possible to generate infeasible solutions?
  - Evaluation function
  - Other mechanisms
    - Heuristics
      - Localsearch
    - Restart
- Once we have made all this decisions we can start the implementation
  - Comment your code
  - Make the output easy to understand
    - And parse!
  - Input parameters by command line
    - Think in the experiments! → make your life easier

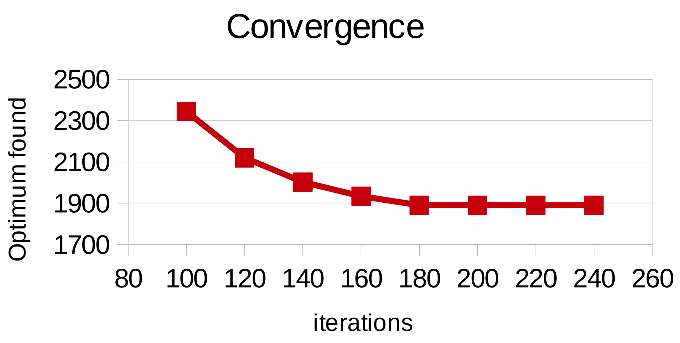


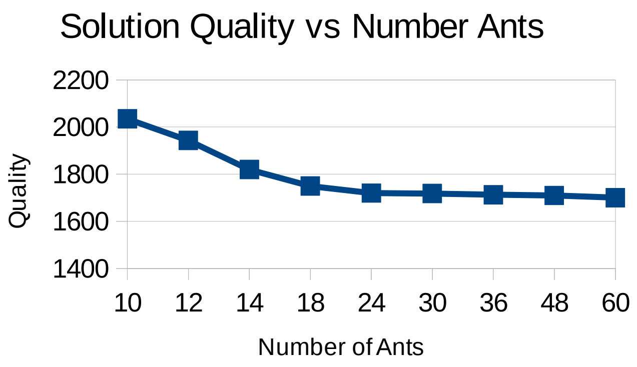
Experiments
Setting parameters

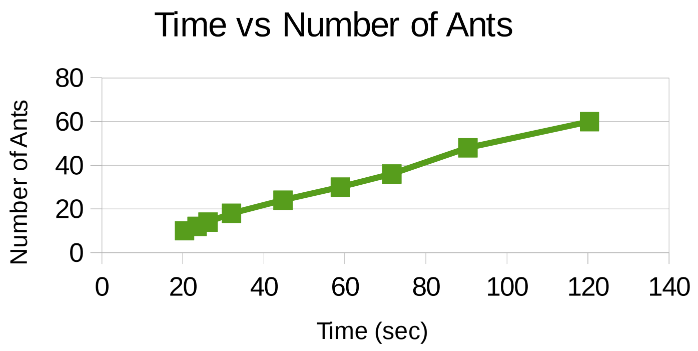
- The behaviour of algorithms depend of the value of their **parameters** and the **instances** being solved.
- Better results can be obtained if we have adequate parameters values.
- Parameter Tuning:
  - Manual
  - Automated
    - Irace
    - ParamILS
    - Smac...
- Identify all the parameters of the algorithms
  - Types
  - Dependencies
  - Values → create a set of “interest” for the problem
- Define a set of representative instances

Experiments
Setting parameters

- Organize your experiments define their goals.
  - Some parameters maybe more important for the algorithm than others
  - In some cases tuning is not needed
  - Parameters can have interactions → think about analysing them together

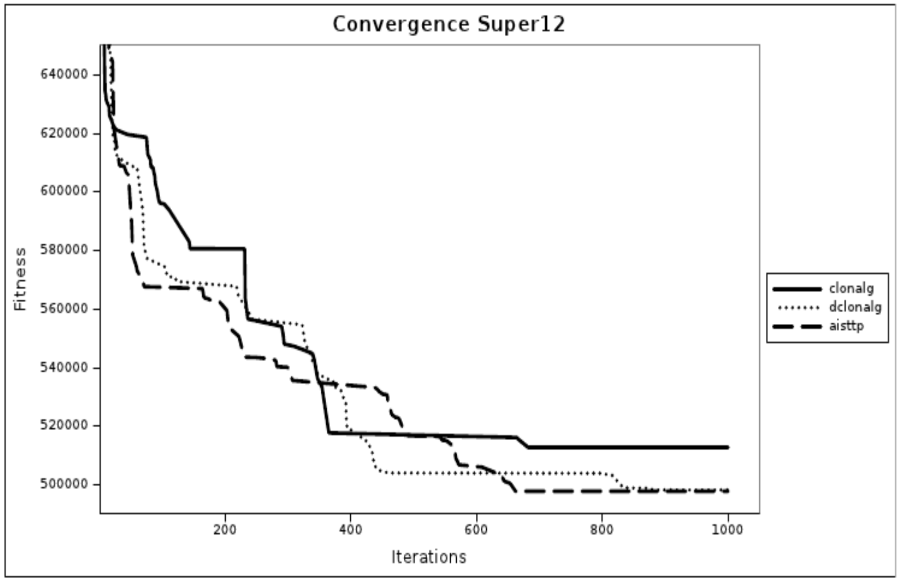


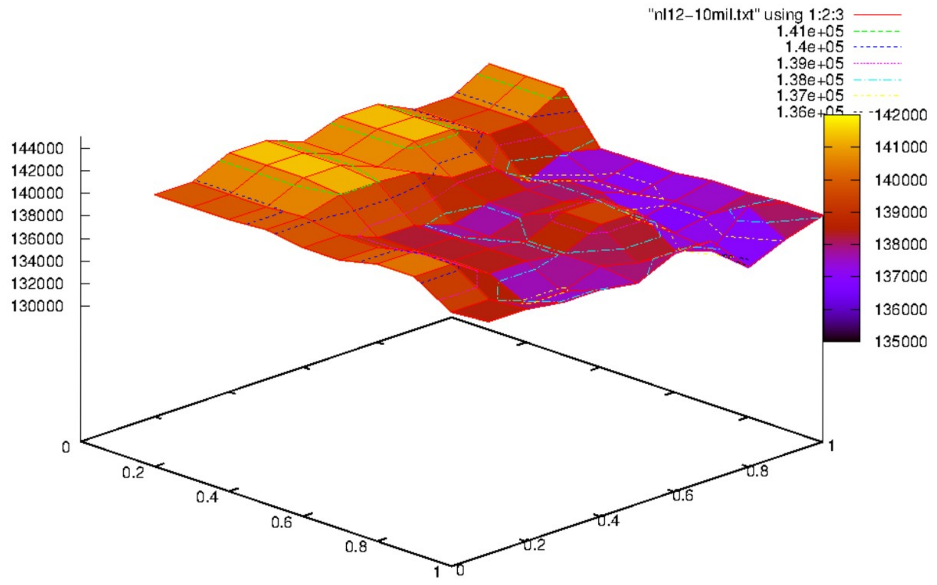


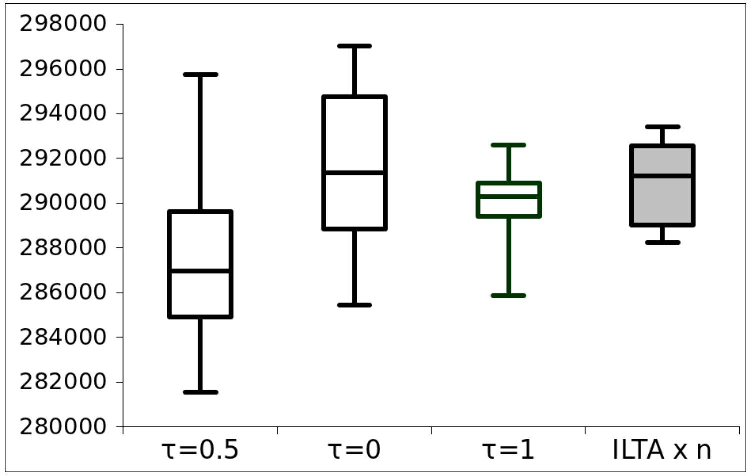


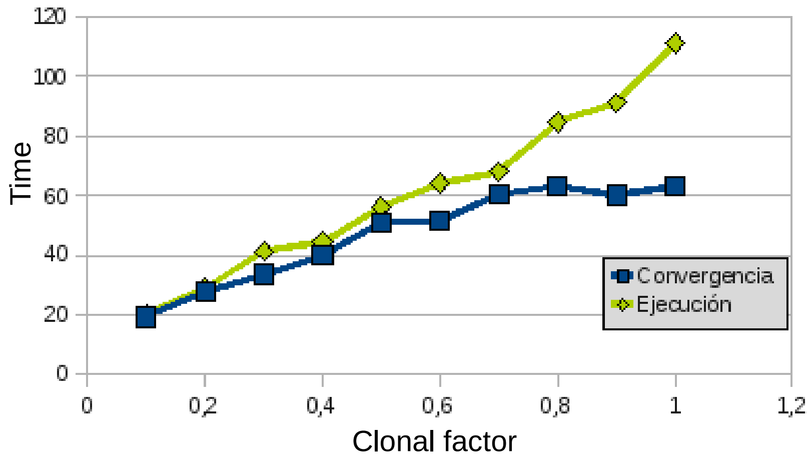
Experiments

Setting parameters example









Experiments
Running experiments

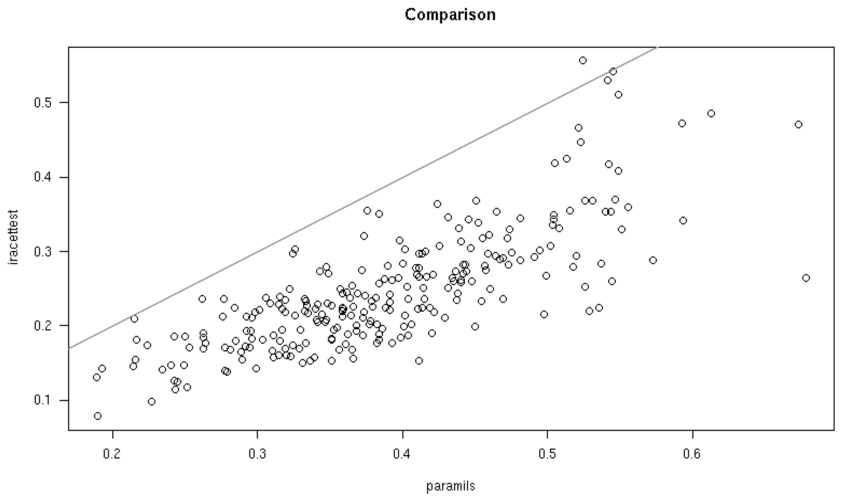
- Set an hypothesis for each experiment and define the way you will try to probe it.
- Define:
  - A Set of testing instances
  - Number of experiments (runs)
  - Stopping criteria (time, iterations, evaluations)
  - Parameters values
- Comparing your algorithm
  - What to compare? → time, quality, others
  - The compared tests should use the same amount of a defined resource in order to be compared, example:
    - Compare quality over
      - 250 sec. of algorithm execution
      - 500 evaluations performed
      - 140 iterations
    - Compare time to reach the optimum value / feasible solution.

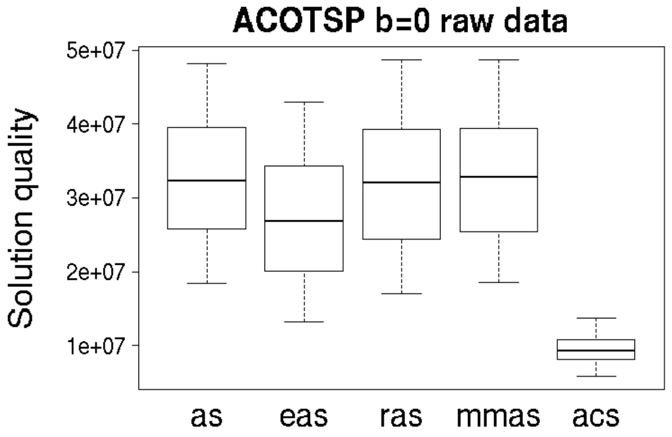
Experiments

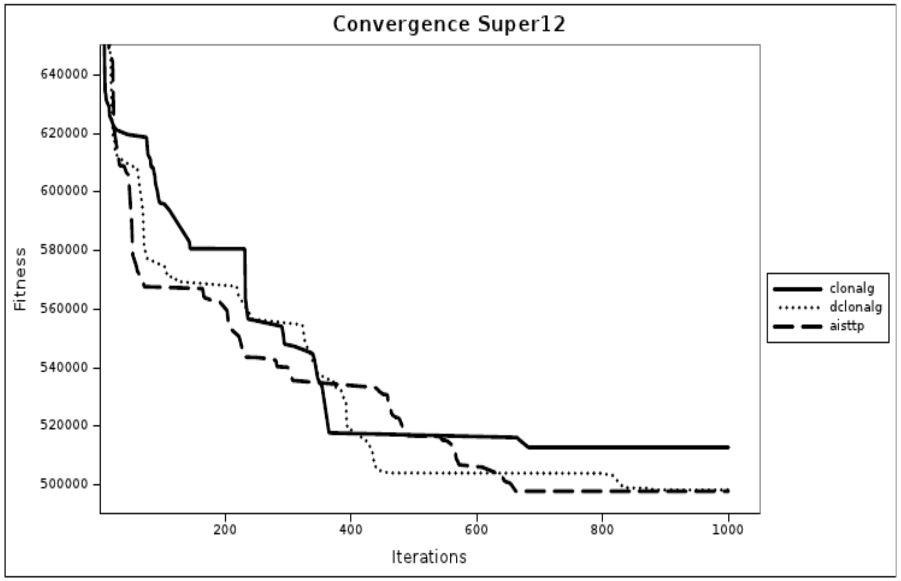
Running experiments

- Comparing your algorithm
  - Define seeds (for stochastic algorithms)
  - When possible:
    - Use statistical tests (T-test, Wilcoxon signed-rank test)
      - R is an statistical software that can compute this tests for you.
    - Use the graph that is more adequate for your purposes
      - Convergence
      - Boxplot
      - Scatter
  - Report:
    - Detailed results and graphs
    - Conclusions about them.

Experiments
Running experiments







Exercise

- We propose to use your expertise in ACO to solve a new problem.
  - Choose a problem from http://people.brunel.ac.uk/~mastjjb/jeb/info.html , or any other you like, ideas:
    - Job shop
    - Graph coloring
    - Vehicle routing problem
    - Variations of TSP
  - Select a small set of instances to make some tests
  - Design an Ant System algorithm for for solving problem
  - Implement your algorithm
  - Analyse its parameters manually and define a good set of values