NOTE ON META-HEURISTIC

G. Antoniol - © 2021 antoniol@mazarol.org

FIRST THINGS FIRST

- Do you really need it?
- Cannot be solved by full search?
- Is there a dynamic programming like solution?
- Is there a polynomial solution?

TWO MAIN FAMILIES

- Local search
  - Hill climbing, simulated annealing, tabu search, …
    - You start with just a single solution and evolve it
    - Limit in the number of iterations or when improvement goes below a threshold

- Evolutionary algorithm
  - One population composed by many individuals (solutions) typically 50, 100 or 200
  - Limit in the number of generations or when improvement goes below a threshold
- Different flavors: mono and multi-objectives
  - GA,
  - NSGASII

WE ALWAYS HAVE

- A solution representation
  - constraints: equality or range
    - variable(s) must be either equal or included in an interval
- A cost/fitness function
- An initialization
- Selection
- Transformations/Modifications/Generation of new individual(s)
- Fitness/cost evaluation
- Evolutionary approaches: duplicate solution detection/removal
  - not always really needed but we strive to have diversity in a population

SOLUTION

- It is the most important element
- Keep it simple but not too simple
  - If possible, use vector(s) of fixed length
  - Prefer simplicity over speed
    - We need to understand what we did even months or years after we did!
- Must be tied to your problem
  - Yes, you can always use bit vectors, but does it make sense?

SOLUTION

- Keep in mind and think very hard about
  - Modern hardware
  - How to best use cash memory
  - How to take advantage of multiple cpu/gpu

- Hard disks, especially ssd(s), are way faster then the network avoid like the hell storing intermediate computation in a NAS
  - If you can !

FITNESS

- Tied to your problem AND your solution
  - It is evaluated over your solution(s)
  - Make code readable, comment variable and code block, classes, etc.
  - Write simple test cases and document them
  - Write simple wrapper to execute simple test cases

- If it is a weighted sum prefer affine transformations aka coefficient add up to 1
  - Be aware a weighted sum may not be the best for a multi-objective

INITIALIZATION

- Often if not always we start with random initial point(s)
  - Random initial solution(s)
  - Knowledge based solutions
    - The closer to be a good solution candidate the better
- If it is a population approach check for duplicates and discard them
- If it violates constraints discard the solution and generate a new one
- Sometimes it must be problem specific to make the algorithm converge in a faster way

INITIALIZATION EXAMPLES

- Suppose we have a bin packing problem or a similar one
  - N objects must be placed into M bins
    - Try to fit as many objects per bin as possible
    - Start with the largest object and add small one

- Suppose we have to define the schedule of a project where people have a skill and a number of hours per week they work
- We know (aprox) how many hours each work package needs
  - Assume there is no overhead if 2 or more works on the same work package
  - Assign the proper skill and enough people to each work package
  - The model is an array where cells represent the people and the number in the cell the team plus we have another vector where work packages are assigned to teams
  - A developer must be in one or mre team and a work package must be assigned to a singke team,

SELECTION

- Local search e.g., hill climbing
  - Pick the only individual
- Evolutionary methods
  - the bests, the top k bests
- Various mechanisms exist
  - roulette wheel
- Decide how many to select is problem dependent aka % of cross-over
  - Cross-over percentage varies largely 20% to 40% is not uncommon
- Do you save/keep the top best – elitism
  - Typical values I used 5 to 10 over 50 to 100 individuals

MUTATION

- Problem dependent
  - Bit flip, value flip, add a queen, delete a queen
  - Swap objects in bins
  - Move developers across teams or assign a task to a different team, add a developer or remove a developer

- It may disrupt constraints
  - Penalize versus repair
  - Mix mutation with local search aka memetic algorithms

CROSSOVER

- It is highly problem and solution representation specific

- Wrong solution representation and/or wrong crossover you are lost
- One point is often enough
  - Keep in mind you MUST preserve solution semantic and consistence
    - See task assignments, test clustering
- Specialized operators do exist
  - For example, in a whole arithmetic crossover, decide in which direction to move the off springs

FITNESS EVALUATION

- Be smart you cannot always re-evaluate all individual
  - Compute fitness only if really needed
  - Track changed individuals
  - Store and recycle fitness

MONO VERSUS MULTI OBJECTIVES

- Sometimes a problem has contrasting objectives
  - Developers effort versus time to finish
  - Number of servers versus time I have to wait test to finish
- When there is a tension between objectives better use a multi-objective approach
  - NSGA-II

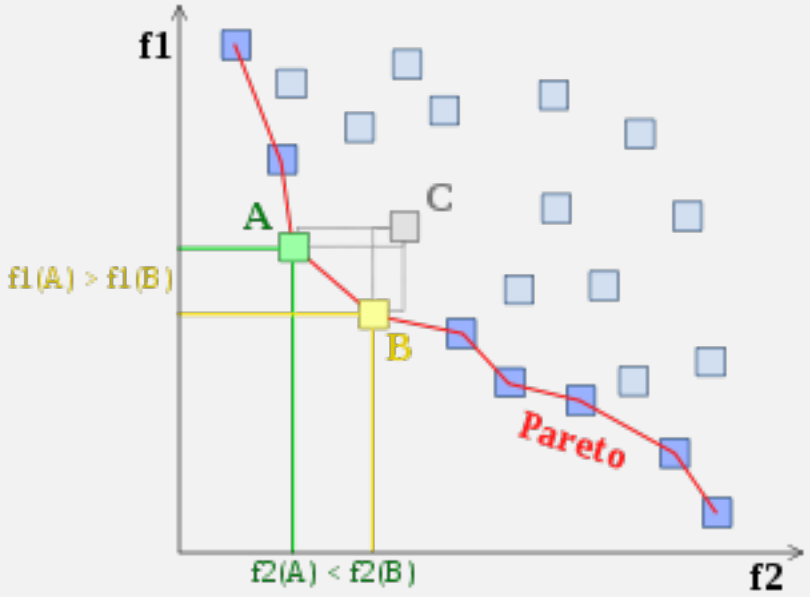

Image from wikipedia

PARETO FRONT

- We seek to find a compromise
  - Cost versus quality
  - Cost versus time
- Are they contrasting goal
  - If yes then use multi-objective approach
- If possible, stay withing 3 dimensions
  - Better 2 but sometime we have to go wild …

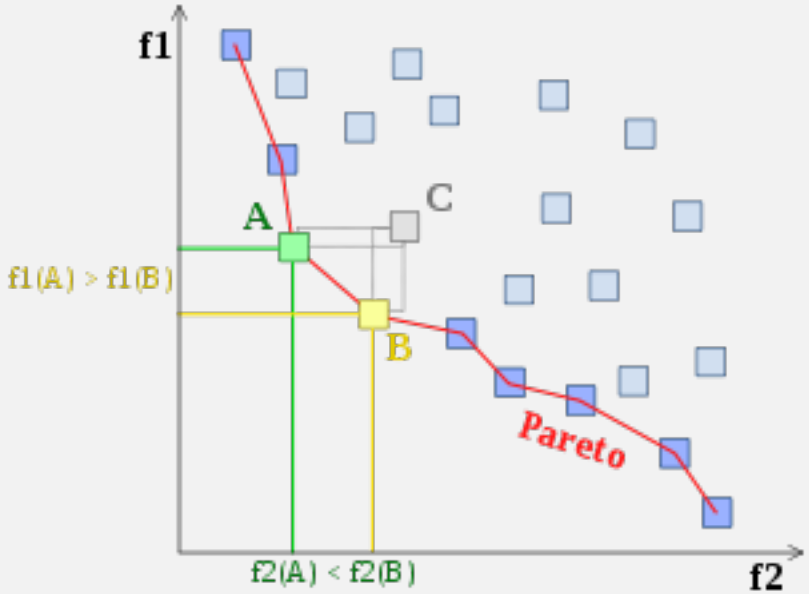

Image from wikipedia

PARETO FRONT

Pareto dominance: set of points for which you cannot improve one dimension without making the other(s) worse

Given x,y ∈ S, x Pareto dominates y

If

$$f_i(x) \leq f_i(y) \ \forall i = 1,\ldots,k \ \text{ y } \exists j \in \{1,\ldots,k\} : f_j(x) < f_j(y)$$

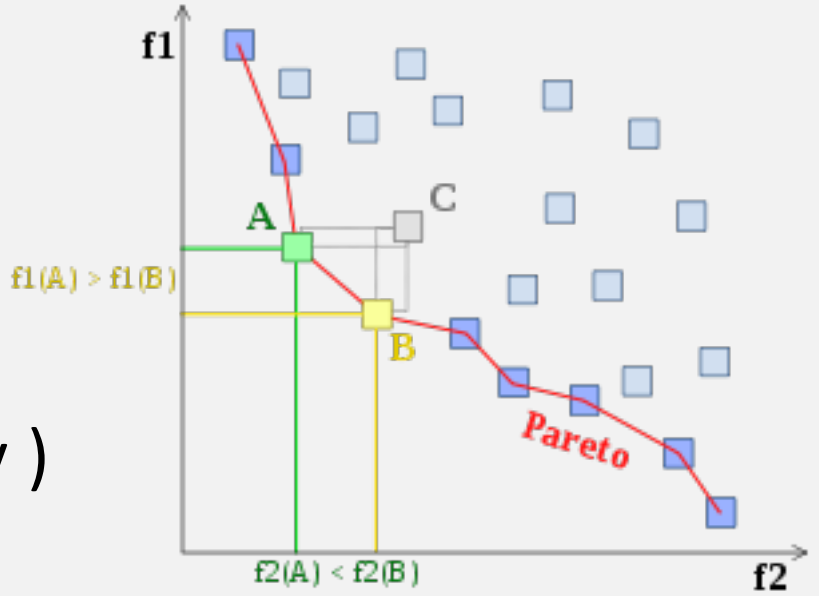

Image from wikipedia

Oftentimes, we seek to get as close to the origin as possible !