UNIVERSITAT POMPEU FABRA

Metaheuristics methods

Helena Ramalhinho

Universitat Pompeu Fabra Barcelona

Outline

- Introduction to Metaheuristics
- Local Search
- GRASP
- Tabu Search
- *Iterated Local Search*
- BRKGA
- Hybrid Metaheuristics
- Matheuristics
- SimHeuristics
- Learnheuristics

Solution Methods

Local Search Methods
Metaheuristics

Integer Programming
Exact Methods

Solution Methods

- Branch-and-bound
- Branch-and-cut
- Column generation
- Cutting and price
- Dynamic programming
- Lagrangean relaxation
- Linear relaxation
- Surrogate relaxation
- Etc…

Integer Programming
Exact Methods

Solution Methods

Local Search Methods
Metaheuristics

- **Local Improvement**
- **Tabu Search**
- **Iterated Local Search**
- Simulated annealing
- **Genetic Algorithms**
- Evolutionary algorithms
- Ant Colony Optimization
- Scatter Search
- Memetic Algorithms
- Etc….

Solution Methods

Local Search Methods
Metaheuristics

- Good solutions for complex and large-scale problems
- Short running times
- Easily adapted

- Mathematical proved optimal solutions
- Important information on the characteristics and properties of the problem.

Integer Programming
Exact Methods

UNIVERSITAT POMPEU FABRA

Algorithms

- Heuristics
  - Any approximate method build up on the basis of the structural properties or on the characteristics of the problem solution, with reduced complexity with respect to exact methods and providing, in general, good feasible quality solutions, without a forma guarantee of solution quality.
  - Cost vs. Time

Algorithms

- Metaheuristics
  - The process of finding a good solution (eventually the optimum) consists in applying at each step (guiding) a subordinate heuristic which has to be design for each particular problem.
    - * C. Ribeiro [1996]
  - Have been designed to attack complex optimization problem where classical heuristics and optimization have failed (so far) to be effective and efficient.

Metaheuristics

- Four attributes of Heuristics and Metaheuristics:
  - Accuracy
    - * Close to optimal
  - Speed
    - * Small computational time
  - Simplicity
    - * No parameters adjustment / easy to program
  - Flexibility
    - * Easy to adapt to other real problems

Cordeau JF, Gendreau M, Laporte G, Potvin JY, Semet F (2002) A guide to vehicle routing heuristics. J Oper Res Soc 53:512–522.

Metaheuristics for real problems

- Metaheuristics
  - Highly effective on hard problems
  - Modularity
  - Easy implementation
  - Short updates
  - Robust
  - Able to give good solutions in short time.
- Real Problems
  - Complex problems
  - Rapid changes in reality
  - Need to quick implementation
  - Different aspects in different sectors
  - Need to quick answer and multiple scenarios

Local optimization algorithms

- Given a solution, attempts to improve this solution by making local modifications at each iteration.
- Neighborhood
  - $N: A \rightarrow 2^A$ subset of feasible solutions
  - Define a local modification, move:
    - * the neighborhood of a solution *x* is the subset of feasible solutions obtained from applying this move to *x*.
  - Local optima solution *x*:
    - * $c(x) \leq c(y)$ for all $y \in N(x)$
- Search Strategy
  - the name of the metaheuristic come usually from the type of search.

Local optimization algorithms

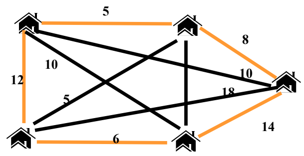
- Local search
  - 1. Get a initial solution x (current solution). Use a constructive heuristic.
  - 2. Search the neighborhood. While there is an untested neighbor of x:
    - * 2.1 Let x' be an untested neighbor of x;
    - * 2.2 If c(x')<c (x) set x=x'; (x' is the new current solution)
  - 3. Return x (local optimal solution).

Local optimization algorithms

- Design of a local optimization algorithm:
  - Obtain an initial solution
    - Heuristic
    - Random solution
    - Constructive method
  - Define the neighborhood
    - Specific for each problem
  - How to search the neighborhood
    - Complete search
    - First improvement

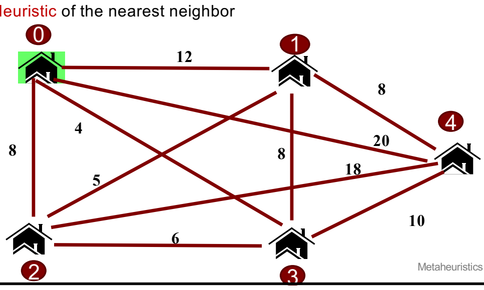
Traveling Salesman Problem

- Traveling Salesman Problem
  - Traveling Salesman Problem
    - E: set of edges, each has a cost c(e);
    - A: any subset of edges forming a Hamilton cycle;
    - c(x): total cost of the edges in x.



Vehicle Routing

- Traveling salesman problem – initial solution
  - Heuristic of the nearest neighbor

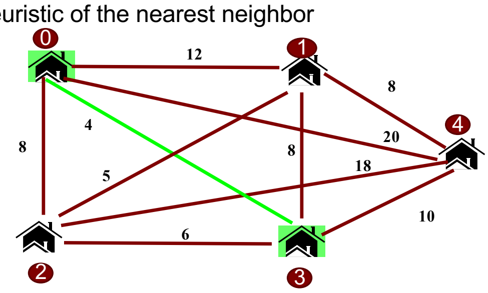


Nearest neighbor algorithm

- Nearest neighbor algorithm on TSP
  1. stand on an arbitrary vertex as current vertex.
  2. find out the shortest edge connecting current vertex and an unvisited vertex V.
  3. set current vertex be V.
  4. mark V as visited.
  5. if all the vertices in domain are visited then terminate. Else go to step 2.

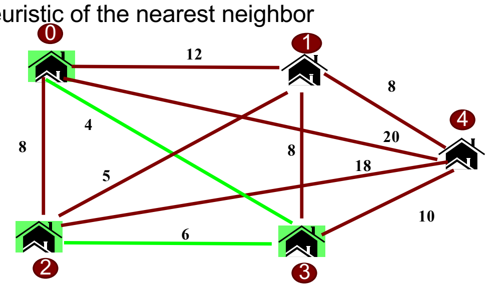
Vehicle Routing

- Heuristic of the nearest neighbor



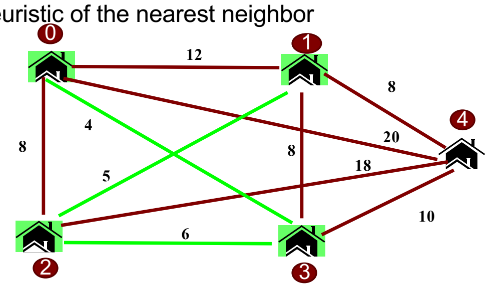
Vehicle Routing

- Heuristic of the nearest neighbor



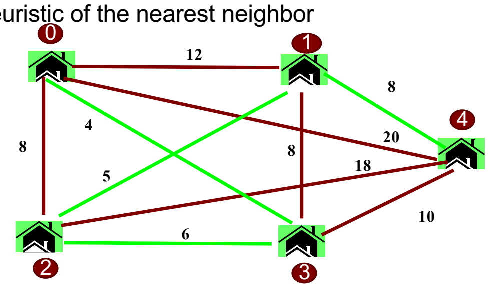
Vehicle Routing

- Heuristic of the nearest neighbor



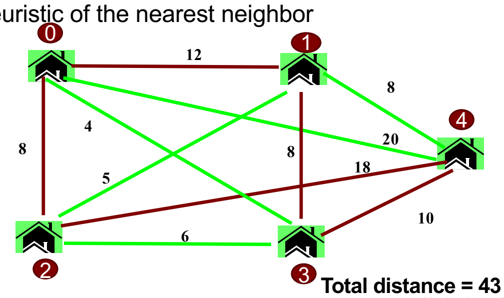
Vehicle Routing

- Heuristic of the nearest neighbor



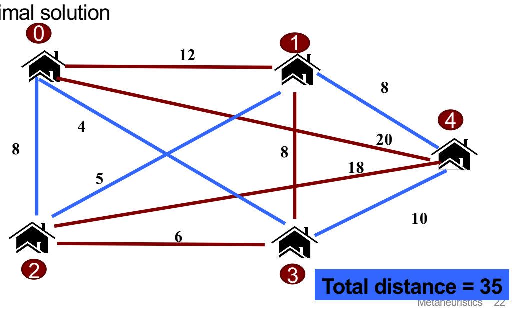
Vehicle Routing

- Heuristic of the nearest neighbor



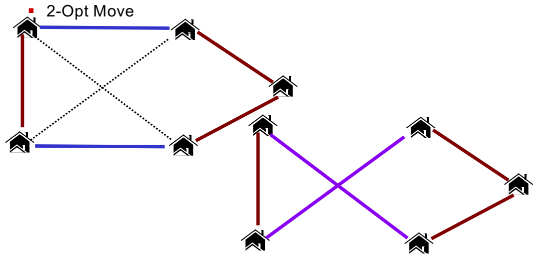
Vehicle Routing

- Optimal solution



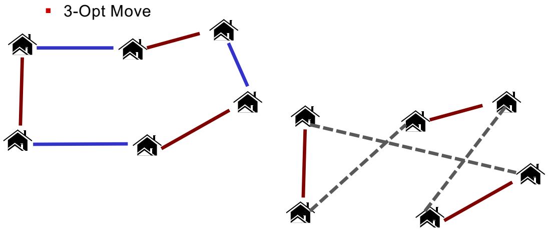
Neighborhood

- Traveling Salesman Problem:
  - 2-Opt Move



Neighborhood

- Traveling Salesman Problem:
  - 3-Opt Move



Local Search

- Examples…
  - Construction heuristics
  - 2-opt local search method
- http://www-e.uni-magdeburg.de/mertens/TSP/TSP.html

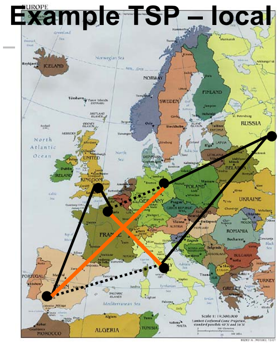
Example TSP – local search application

- Initial solution and current solution
- Nearest neighbor heuristic

The closest vertex to Berlin is Paris
The closest vertex to Paris is Londres
The closest vertex to Londres is Sevilla
The closest vertex to Sevilla is Roma
The closest vertex to Roma is Moscow
The nearest to the initial vertex is Moscow
SOLUTION: Nearest-Neighbor

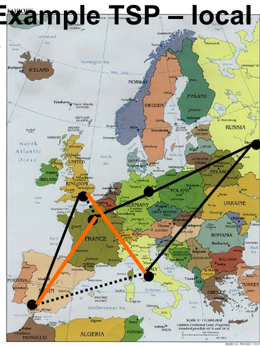
Route: Berlin, Paris, Londres, Sevilla, Roma, Moscow, Berlin
Length: 65

Example TSP – local search application



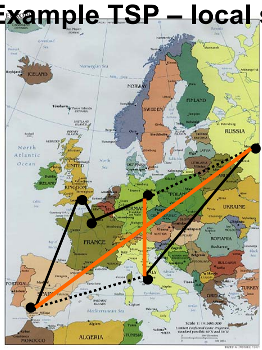
- 2-opt neighborhood
  - 9 neighbors
  - Neighbor 1
  - Cost: 69

Example TSP – local search application



- 2-opt neighborhood
  - 9 neighbors
  - Neighbor 2
  - Cost: 69

Example TSP – local search application



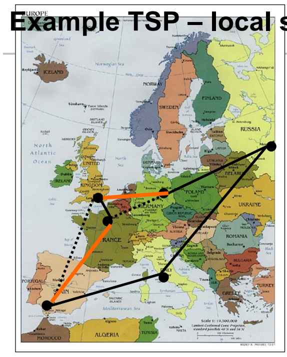
- 2-opt neighborhood
  - 9 neighbors
  - Neighbor 3
  - Cost: 78

search application

- 2-opt neighborhood
  - 9 neighbors
  - Neighbor 4
  - Cost: 67

Universitat Pompeu Fabra Barcelona

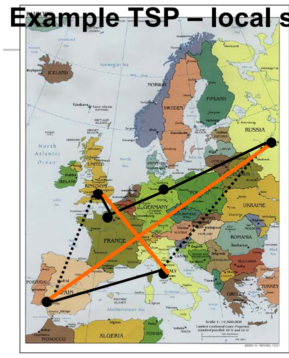
Example TSP – local search application



- 2-opt neighborhood
  - 9 neighbors
  - Neighbor 5
  - Cost: 63

Universitat Pompeu Fabra Barcelona

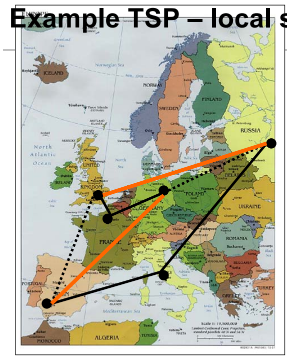
Example TSP – local search application



- 2-opt neighborhood
  - 9 neighbors
  - Neighbor 6
  - Cost: 73

Universitat Pompeu Fabra Barcelona

Example TSP – local search application



- 2-opt neighborhood
  - 9 neighbors
  - Neighbor 7
  - Cost: 76

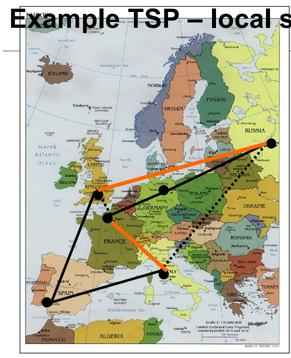
Universitat Pompeu Fabra Barcelona

Example TSP – local search application

- 2-opt neighborhood
  - 9 neighbors
  - Neighbor 8
  - Cost: 78

Universitat Pompeu Fabra Barcelona

Example TSP – local search application



- 2-opt neighborhood
  - 9 neighbors
  - Neighbor 9
  - Cost: 69

Universitat Pompeu Fabra Barcelona

E… search application

- Best neighbor (5)
- Cost 63
- New current solution
  - Repeat until find a local optimal
  - Local optimal solution: The current solution is better than any solution in the neighborhood.

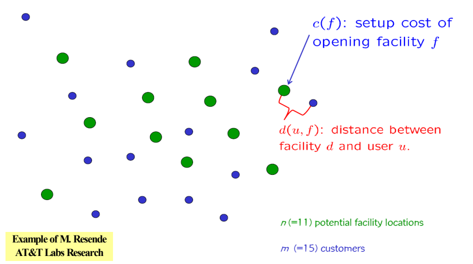
Universitat Pompeu Fabra Barcelona

Facility Location Models

- Important problem in logistics
  - Where to locate new facilities.
    - Retailers, warehouses, factories.
  - Very complex problems
- Warehouse location problem
  - to locate a set of warehouses in a distribution network
- Cost of locating a warehouse at a particular site:
  - fixed cost vs variable cost
    - cost of open facility vs. transportation cost

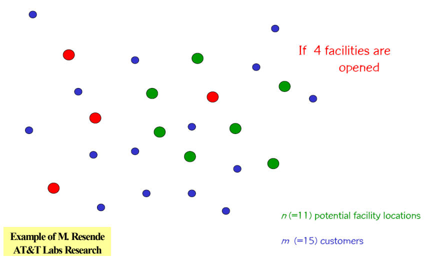
Universitat Pompeu Fabra Barcelona

Location example – Local search



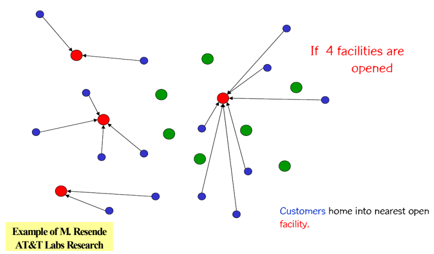
Universitat Pompeu Fabra Barcelona

Location example – Local search



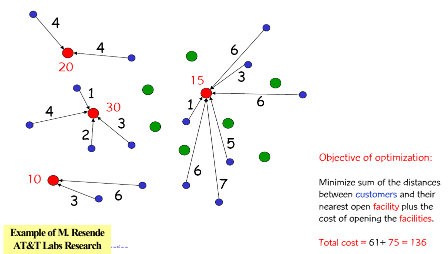
Universitat Pompeu Fabra Barcelona

Location example – Local search



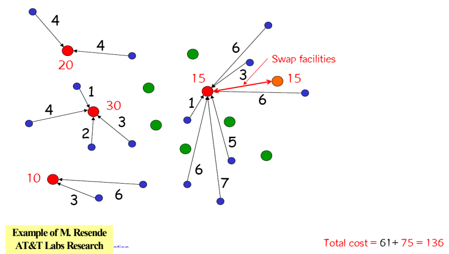
Universitat Pompeu Fabra Barcelona

Location example – Local search



Universitat Pompeu Fabra Barcelona

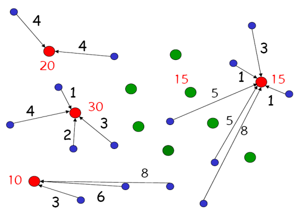
Location example – Local search



UNIVERSITAT POMPEU FABRA

Universitat Pompeu Fabra Barcelona

Location example – Local search

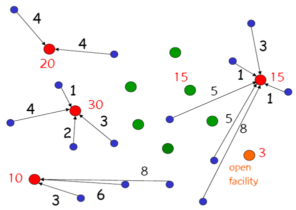


Example of M. Resende
AT&T Labs Research

Total cost = 58 + 75 =
133 < 136

Universitat Pompeu Fabra Barcelona

Location example – Local search

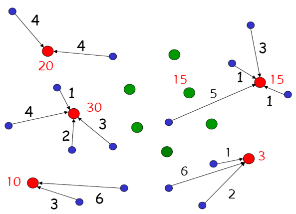


Example of M. Resende
AT&T Labs Research

Total cost = 58 + 75 =
133 < 136

Universitat Pompeu Fabra Barcelona

Location example – Local search

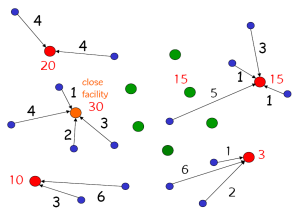


Example of M. Resende
AT&T Labs Research

Total cost = 46 + 78 =
124 < 133

Universitat Pompeu Fabra Barcelona

Location example – Local search

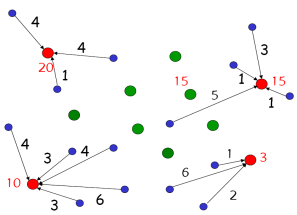


Example of M. Resende
AT&T Labs Research

Total cost = 46 + 78 =
124 < 133

Universitat Pompeu Fabra Barcelona

Location example – Local search



Example of M. Resende
AT&T Labs Research

Total cost = 48 + 48 =
96 < 124

Universitat Pompeu Fabra Barcelona

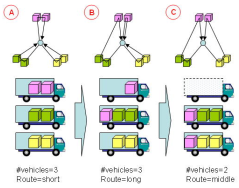
Vehicle Routing

- A set of customers at known geographical locations has to be supplied by a fleet of vehicles from a single depot.
- Each customer has a specific demand.
- Each route starts and finish at the depot.
- The objective is to find the set of routes whose total length or cost is minimal.
  - Or minimize the number of routes.

Vehicle Routing

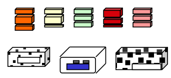
Careful with the vehicle capacity!

- Local search
- Moves:
  - 2-opt like in TSP.
  - Interchange customers in routes.



The Job-Shop Scheduling

- A set of *n* jobs, J_1, ... , J_n
- A set of *m* machines, M_1, ... , M_m
- Schedule the processing of each job by the machines.
- The objective is to minimize the maximum completion time (makespan)
  - Min C_{max}= max C_j.



The Job-Shop Scheduling

- Initial solution method
- Priority dispatching rule:
  - Consider a priority function.
  - The first operations of all jobs are inserted in a priority queue.
  - At each iteration:
    - Schedule the maximum priority operation as soon as possible in the correspondent machine.
    - Delete it from the queue.
    - The next operation of the same job, if exists, is inserted in the priority queue.
- Other approaches: Shifting Bottleneck Heuristic

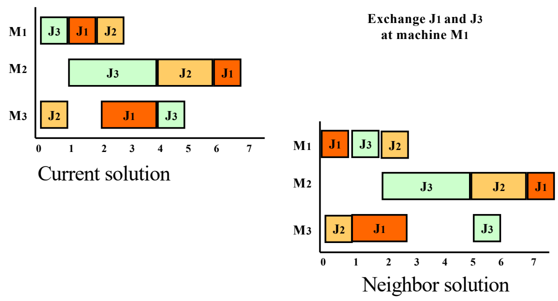
Local search for the Job-Shop

- Local Search Method
  - Initial schedule
  - Simple Neighborhoods:
    - van Laarhoven, Aarts & Lenstra (1992)
  - More sophisticates neighborhood and metaheuristics:
    - Dell'Amico and Trubian (1993)
    - Nowicki, E. and C. Smutnicki (2005)

Neighborhood for Job-Shop

- Neighborhood
  - van Laarhoven, Aarts and Lenstra (1992)
  - Exchange the processing order of two operations.
    - Reverse an arc in the graph representing the solution.
  - Successive operations processed in the same machine.
  - Critical operations.
    - Below to the longest path in the graph.

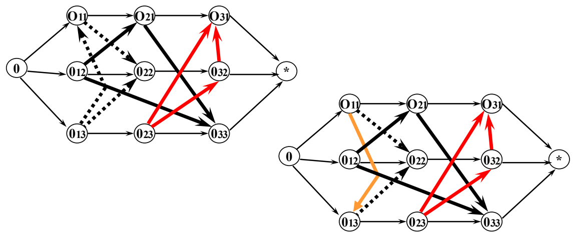
Neighborhood for Job-Shop



UNIVERSITAT POMPEU FABRA

Neighborhood for Job-Shop

- Reverse an arc



Local Search Software

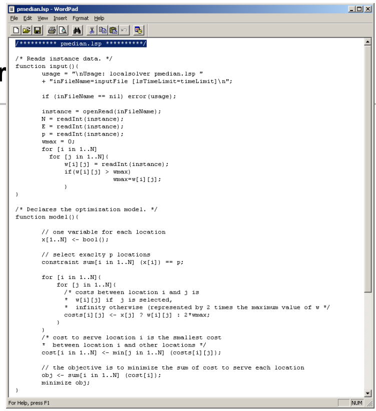
- Local Solver
  - http://www.localsolver.com/

Local Search Software

- Local Solver
  - http://www.localsolver.com/
  - Resulting from a 5-year R&D project involving distinguished experts in computational optimization, LocalSolver is the first math programming software combining the simplicity of use of a model-and-run solver and the power of local-search techniques for combinatorial optimization. LocalSolver includes an innovative math modeling language for fast prototyping and lightweight object-oriented APIs for full integration (C++, Java, .NET).

Local Sear

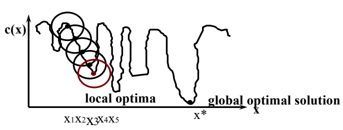
- Example pmedian




```
/********** pmedian.lsp **********/

/* Reads instance data. */
function input(){
       usage = "\nUsage: localsolver pmedian.lsp "
       + "inFileName=inputFile [lsTimeLimit=timeLimit]\n";

       if (inFileName == nil) error(usage);

       instance = openRead(inFileName);
       N = readInt(instance);
       E = readInt(instance);
       p = readInt(instance);
       wmax = 0;
       for [i in 1..N]
         for [j in 1..N]{
             w[i][j] = readInt(instance);
             if(w[i][j] > wmax)
                         wmax=w[i][j];
             }
}

/* Declares the optimization model. */
function model(){

       // one variable for each location
       x[1..N] <- bool();

       // select exaclty p locations
       constraint sum[i in 1..N] (x[i]) == p;

       for [i in 1..N]{
           for [j in 1..N]{
             /* costs between location i and j is
              *  w[i][j] if  j is selected,
              *  infinity otherwise (represented by 2 times the maximum value of w */
             costs[i][j] <- x[j] ? w[i][j] : 2*wmax;
           }
       }
       /* cost to serve location i is the smallest cost
        *  between location i and other locations */
       cost[i in 1..N] <- min[j in 1..N] (costs[i][j]);

       // the objective is to minimize the sum of cost to serve each location
       obj <- sum[i in 1..N] (cost[i]);
       minimize obj;
}
```


Local optimization algorithms

- Comments
  - The search stops at the first local optimum solution with respect to the neighborhood N.
  - The final solution highly depends on the initial solution and on the neighborhood.
  - No way back out of unattractive local optima...



Computational Results

- Local search is clearly inferior to other metaheuristics.
  - Too many local optimal solutions
- Tabu-search outperformed the other methods.
- Iterated Local Search
  - Large-step optimization combined with tabu-search gave the better results.

UNIVERSITAT POMPEU FABRA

Multi-Start / Re-start Heuristics

- Iterative improvement or hill-descending
  - 1. Get a initial random solution x.
  - 2. Run a **Local Search Method** (output x)
  - 3. If cost(x)<cost(x_{best}) set x_{best}=x;
  - 4. If the stop criteria is not verified, go back to step 1.
  - 5. Output the best solution found.
    - Comments
      - Successive repetition of local improvement.
      - Easy to implement.
      - Random solutions may be very bad.

GRASP

- Greedy Randomized Adaptive Search Procedure
  - 1. While stopping criterion not satisfied:
    - 2. Get a **greedy randomized initial** solution x.
    - 3. **Local Search** starting with solution x
  - 4. Output the best solution found.
- Greedy randomized solution
  - 1. While solution not complete:
    - 2. Using a greedy criterion, make a candidate list of elements to enter the solution.
    - 3. **Select one element randomly from the candidate list.**
    - 4. Insert it in the solution.

GRASP

- Comments
  - Very easy to implement
  - Greedy heuristics can be develop for almost any combinatorial problem.
  - Very few parameters to be set.
  - Many successful applications.
- References
  - M.G.C. Resende and C.C. Ribeiro (2017) "Greedy randomized adaptive search procedures: Advances and extensions"
  - http://mauricio.resende.info/papers.html

Bias Ramdomization Heuristics

- Biased Randomization of Heuristics
  - Introduce of a slight modification in the greedy constructive behavior that provides a certain degree of randomness while maintaining the logic behind the heuristic.
  - BRPs can be categorized into two main groups according to how choice probabilities are computed:
    - BRPs using an empirical bias function;
    - BRPs using a skewed probability distribution.

Grasas A., Juan, A.A. Faulin, J., De Armas, J. and **Ramalhinho H.** (2017), Biased Randomization of Heuristics using Skewed Probability Distributions: applications to routing and other problems, *Computers & Industrial Engineering* 110: 216–228. Doi: 10.1016/j.cie.2017.06.019.

MIRHA

- Multi-start biased randomization of heuristics with adaptive local search
  - Apply a Greedy Classical Heuristics randomized using a bias distribution:
    - The main idea of these heuristics is to select the next step from a list of available movements, usually according to a greedy criterion.
    - we consider non-uniform and nonsymmetric (biased) distributions, e.g.: the geometric distribution or the decreasing triangular distribution.
  - Local Search

MIRHA

- procedure MIRHA(inputData, endConditions, prob.Dist., seed, heuristic)
  - initializeRandomGenerator(seed);
  - while endCondition[1]= false do
    - solution = getRandomSolution(inputData, heuristic, prob.Dist.);
    - solution = adaptiveLocalSearch(solution, endCondition[2]);
    - bestSolution = updateBestSolution(solution);
  - end while;
  - return bestSolution;
- end MIRHA

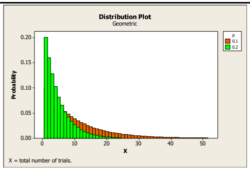
UNIVERSITAT POMPEU FABRA

Universitat Pompeu Fabra Barcelona

Bias Ramdomization ILS

- Skewed Theoretical Probability Distributions
  - Pseudocode to select the next element using a skewed distribution.

```
Procedure BRP(L, s, PD, p)
    μ ← using seed s, generate pseudo-random number in [0,1)
    ρ ← using μ, generate random variate from distribution PD(p)
    l ← select the ρ-th element of the sorted list L
    return l
End
```



Universitat Pompeu Fabra Barcelona

Simulated Annealing

- Simulated Annealing
  - Attempts to avoid stopping in bad local optima by allowing uphill moves.
  - The probability of accepting uphill moves decreases with the number of iterations performed.
  - Convergence proof (Markov Chain)
  - Easy to implement
    - implementation choices: problem-specific and generic.
  - Large computation running times
    - Kirkpatrick, Gelatt & Vecchi [1983], Cerny [1985]
    - Johnson et al. [1989]

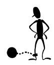
Universitat Pompeu Fabra Barcelona

Simulated Annealing

- Simulated annealing with a geometric cooling schedule:
  - Get a initial solution x;
  - Get a initial temperature t;
  - While a certain stop-criteria is not verified:
    - Perform the following loop L times:
      - Pick a random neighbor x' of x;
      - Let dc= cost(x')-cost(x)
      - If dc ≤ 0 set x=x',
      - otherwise set x=x' with probability exp(-dc/t);
    - Set t=r*t;
  - Return the best solution found.

Universitat Pompeu Fabra Barcelona

Tabu-Search



- Tabu Search
  - Attempts to escape from local optima and continue the search further.
  - Make the best move found…
    - Problem: alternates between two solution.
  - Tabu list
    - Information about the most recently made moves is kept in a tabu list.
    - Some solution in the neighborhood are tabu for some iterations (tabu-tenure)
  - Adaptive memory
- Glover [1989], Glover [1990], Glover & Laguna [1997]

Universitat Pompeu Fabra Barcelona

Tabu-Search

- Aspiration Criteria
  - Accept a tabu-move if leads to the best solution seen so far.
- Intensification Strategy
  - Focus the search into more promising regions.
    - After a certain number of iterations without an improvement in the best solution, restart the algorithm with the best solution found.
- Diversification strategy
  - Drive the search to unexplored regions.
- Stopping Criteria
  - Maximum number of iterations or
  - One intensification phase where the best solution was not changed.

Universitat Pompeu Fabra Barcelona

Tabu-Search

- Tabu-Search with diversification / intensification strategy:
  - Get a initial solution x;
  - While a certain stop-criteria is not verified:
    - Get the neighbor x' of x, **not tabu** or satisfying an aspiration criteria with minimal cost among the neighbors of x;
    - Set x=x' and **update the tabu list** and aspiration criteria;
  - Execute a diversification or an intensification strategy and repeat.
  - Return the best solution found.

UNIVERSITAT POMPEU FABRA

Tabu-Search

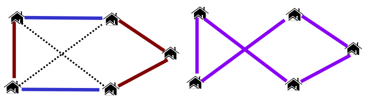
- Design of a Tabu-Search algorithm (1/2):
  - Obtain an initial solution
    - Heuristic
    - Random solution
  - Define the neighborhood
    - Specific for each problem
  - How to search the neighborhood
    - Complete search
    - First improvement

Tabu-Search

- Design of a Tabu-Search algorithm (2/2):
  - Define the tabu criteria
  - Define the aspiration criteria
  - Intensification Strategy
  - Diversification strategy
  - Stopping Criteria
    - Running time
    - Number of iterations without improvement
- All of the above are problem-specific.

Tabu List

- Traveling Salesman Problem: 2-Opt move
  - The tabu list consists of the shortest edged of the two deleted by a move, Glover [1986]
    - tabu move: reinsert this edge
  - The tabu list consist of pairs of edges, each pair being the set of edges deleted in some previous iteration, Knox [1994].
    - tabu move: reinsert this pair of edges

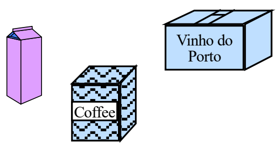


Tabu-Search

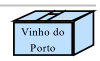
- Comments
  - Produces good solutions in relatively small computational times.
  - Good quality solutions: best solutions for many problems.
  - Large number and diversification of applications.

Production Scheduling Problem in a Cardboard Box Factory

- Factory that produces different cardboard boxes.
  - each order passes first in a trim machine, where the cardboard is cut in rectangles (phase 1).
  - then, goes to several converting machines that finish the box (phase 2) (unrelated parallel machines):
    - Folding
    - Printing
    - Glue
    - Staple



Production Scheduling Problem in a Cardboard Box Factory



- Phase 1
  - A cutting-stock problem.
    - Linear programming
  - The orders are cut with the objective of minimizing the left overs.
- Phase 2
  - Each converting machine can do some specific operations;
  - Each order (job) needs some operations.
  - Release dates and due dates.
  - The processing time of each operation depends on the order and on the machine.
  - The set-up time depends on the machine and operations.

Production Scheduling Problem

- The usual constraints for the generalized job-shop scheduling problem, and...
  - Each operation O_{ij} can be processed by a subset of the machines, but has a preferred one.
  - Each operation Oij has a different processing time in each of the machines, p_{ijk}.
  - Each job has a release time, r_j .
  - Each job has a duedate, d_j .
  - Between the processing of two operations in the same machine there can exist a set-up time.
- Minimize a general cost function.

Metaheuristics 79

A Generalized Job-Shop

- Objective:
  - Minimize a general cost function
- The cost function is a combination of several functions:
  - cost of a machine being idle;
  - cost of the processing an operation in a certain machine;
  - cost of switching from a operation to other operation in a certain machine;
  - cost related with late jobs.

Metaheuristics 80

Production Scheduling

- Two types of moves:
  - Type I:
    - Pick two operations that are processed consecutively in the same machine.
    - Reverse the order of processing these operations.
      - Classical job-shop: only pair of operations in the critical path.
  - Type II:
    - Pick an operation that can be processed in more than one machine.
    - Switch the operation from the actual machine to other machine that can processed it.
  - Switch from one type of move to other after a certain number of iterations, or if there is no improvement.

Metaheuristics 81

Production Scheduling

- Tabu-Search Algorithm
  - Initial solution: Priority dispatching rule (slack time)
  - Moves: Type I &Type II, Tabu List
  - Aspiration Criteria
    - Over rule a tabu move if lead to a better solution seen so far.
  - Intensification Strategy
    - Focus on the scheduling of one machine if this can improve significantly the solution.
    - Use a constructive and improvement heuristic.

Metaheuristics 82

Production Scheduling

- Good results
  - improves the cost of the initial solution about 3% to 7 %.
  - no optimal cost or other solutions methods to compare.
- Flexible method
  - permits to change the cost function without changing the full program.
  - different parameters (tabu-list size, etc.) do not make a big influence in the results.

Metaheuristics 83

Iterated Local Search

- A Local Search Method.
- Combines local optimization with a big transition/perturbation.
  - Diversification strategy based on the structure of the problem.
  - ***Optimization techniques***
- Local optimal solutions.
- Able to make large changes at any stage of the algorithm.

Next Topic!

Metaheuristics 84

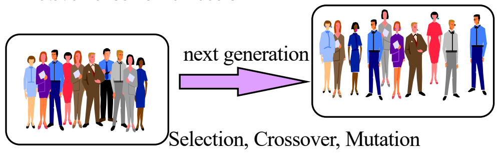
UNIVERSITAT POMPEU FABRA

Genetic Algorithms

- Genetic algorithms
  - Based on evolution
  - Use information of a population of individuals (solutions) during the search for better solutions.
    - Not only information from a single individual.
- Important aspects
  - Representation of the solutions
    - binary: genetic algorithms
    - vector of real numbers: evolution strategies
  - Generation of the initial solution
  - Selection of suitable parents
  - Genetic Operators: crossover and mutation.

Genetic Algorithms

- Main Operators
  - Evaluation of the fitness (objective function) of each individual
  - Selection of well adapted individuals
  - Crossover (recombination) of select individuals
  - Mutation of some individuals



Genetic Algorithms

- (Simple) Genetic Algorithm
  - Get a initial Population P(0);
  - While a certain stop-criteria is not verified:
    - Evaluate the fitness of each individual in P(t);
    - Construct new population P(t+1):
      - Select elements from P(t);
      - Crossover these elements;
      - Mutation of some elements.
  - Return the best solution found.

Bias Random Key Genetic Algorithm

- GAs and random keys
  - Introduced by Bean (1994) for sequencing problems.
  - Individuals are strings of real-valued numbers (random keys) in the interval [0,1].

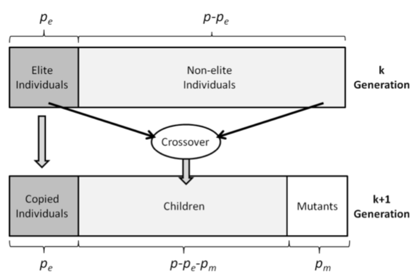
    S = (0.25, 0.19, 0.67, 0.05, 0.89)
    s(1) s(2) s(3) s(4) s(5)
  - **Decode by sorting vector of random keys**
  - Sorting random keys results in a sequencing order:

    S' = (0.05, 0.19, 0.25, 0.67, 0.89)
    s(4) s(2) s(1) s(3) s(5)
    Sequence: 4 – 2 – 1 – 3 – 5

Bias Random Key Genetic Algorithm

- Random-keys vs biased random-keys
  - How do random-key GAs (Bean, 1994) and biased random-key GAs differ?
    - A random-key GA selects both parents at random from the entire population for crossover: some pairs may not have any elite solution
    - A biased random-key GA always has an elite parent during crossover
    - Parametrized uniform crossover makes it more likely that child inherits characteristics of elite parent in biased random-key GA while it does not in random key GA (survival of the fittest)

Bias Random Key Genetic Algorithm



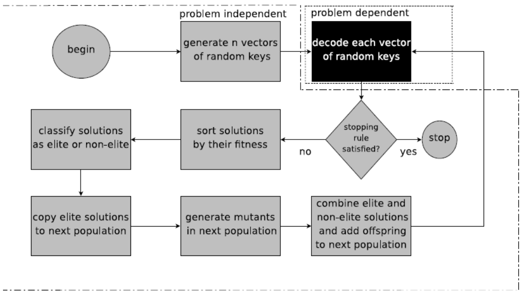
Bias Random Key Genetic Algorithm

Elite parent	0.27	**0.89**	**0.73**	**0.11**	0.55
Non-elite parent	**0.36**	0.98	0.21	0.08	**0.62**
CROSSOVER					
Random values < 0.6? (Probability = 0.6)	(0.86 ,	0.42 ,	0.33 ,	0.19 ,	0.66)
Child	0.36	0.89	0.73	0.11	0.62

Bias Random Key Genetic Algorithm

- J.F. Gonçalves and M.G.C. Resende (2016) “Biased random-key genetic programming” Handbook of Heuristics, R. Martí, P.M. Pardalos, and M.G.C. Resende, eds., Springer, 2016
  - http://mauricio.resende.info/papers.html

Bias Random Key Genetic Algorithm

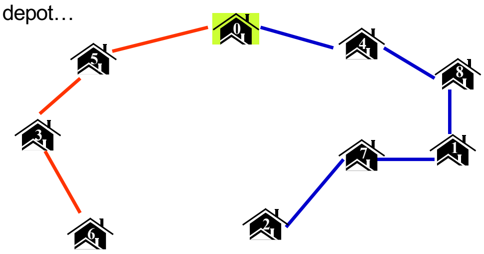


Bias Random Key Genetic Algorithm

- Decoders
  - A decoder is a deterministic algorithm that takes as input a random-key vector and returns a feasible solution of the optimization problem and its cost.
  - Bean (1994) proposed decoders based on sorting the random-key vector to produce a sequence.
  - A random-key GA searches the solution space indirectly by searching the space of random keys and using the decoder to evaluate fitness of the random key.

Open Vehicle Routing

- Visit all customer but does not have to start or finish at the depot…



Open Vehicle Routing

- Time-Constraint Capacity Open VRP (1/2)
  - A direct graph $G = (V, A)$ is given, where $V = \{0, 1, \ldots, n\}$ is the set of $n + 1$ nodes and A is the set of arcs.
  - Node 0 represents the depot while the remaining nodes $V' = V \setminus \{0\}$ corresponds to the n visiting points.
  - Each collection point $i \in V'$ has q_i products/boxes to be transported to the depot (assume $q_0 = 0$).
  - Distance and travel times between each node.

UNIVERSITAT POMPEU FABRA

Open Vehicle Routing

- Time-Constraint Capacity Open VRP (2/2)
  - Open routes (start at the first point and finish at the depot)
  - The vehicle fleet is composed M = {1,…, m} identical vehicles with capacity Q_k.
  - The travel maximum time between the first point to the depot is k hours.
  - **Minimize the total distance (or transportation costs)**

Open Vehicle Routing

- Applications of the Time-Constraint Capacity Open VRP
  - Blood Collection Sample at a Clinical Laboratory (2 hours)
  - Patients transportation to medical exams (1 hour)
  - School Bus (1 hour time constraint)
  - Retailing with subcontracted distribution transportation (8 hours working time)
  - Etc.

Open Vehicle Routing

- Decoder for the Time-Constraint Capacity Open VRP
  - Suppose each collection points has one bag and the capacity of the vehicles is 2.
  - A solution: S = (0.25, 0.19, 0.67, 0.05, 0.89)

    Decoding…

    S'= (0.05, 0.19, 0.25, 0.67, 0.89)

    s(4) s(2) s(1) s(3) s(5)

    Sequence: 4 – 2 – 1 – 3 – 5
  - Means that 3 routes are obtained:
    * 4-2-lab
    * 1-3-lab
    * 5-lab

Open Vehicle Routing- Real Application

- The Blood Sample Collection at a Clinical Laboratory
  - Application Lab 1: 43 collection points
  - Actually 10 routes
  - Data:
    * Laboratory
    * Distances and time matrix by google maps
      – vrp.upf.edu
  - Two scenarios & two different truck capacities

catlab

Open Vehicle Routing- Real Application

- The Blood Sample Collection at a Clinical Laboratory
  * Lab 1: saving 30% of total annual routing costs (around 45.000€)
  * Better management if there are new collection points or changes in the address.
  * Better service quality (2 hours transportation).
  * Better planning in the case of laboratory merging strategy.

Grasas A., Ramalhinho H., Pessoa L.S., Resende M.G.C., Caballé I. and Barba N. (2014) On the Improvement of Blood Sample Collection at Clinical Laboratories, BMC Health Services Research, 14:12 DOI:10.1186/1472-6963-14-12. WoS, 2014 JCR IF = 1.712, Q2. Open access article.

catlab

Open Vehicle Routing- Real Application

- BRKGA
  - The results were obtained in 100 seconds.

Number of Routes				
		Current Solution	BRKGA Solution	
			Vehicle Capacity = 16	Vehicle Capacity = 25
Scenario I	CST+HUMT	5	3	3
	ICS	5	5	4
Scenario II	CST+HUMT+I		7	7

Open Vehicle Routing- Real Application

- Upper and lower bounds on the number of routes
  - using CPLEX
  - 1h (CST+HUMT)
  - 2h (ICS)
  - 3h (CST+HUMT+ICS)

		CPLEX Solution	
		Vehicle Capacity = 16	Vehicle Capacity = 25
Scenario I	CST+HUMT	4(3)	4(2)
	ICS	5*	6(3)
Scenario II	CST+HUMT+I	13(6.43)	14(5)

- Recall BRKGA results in 100 seconds

Number of Routes				
		Current Solution	BRKGA Solution	
			Vehicle Capacity = 16	Vehicle Capacity = 25
Scenario	CST+HUMT	5	3	3
	ICS	5	5	4
Scenario	CST+HUMT+ICS		7	7

Ant Colonies

- Ant Colony Optimization studies artificial systems that take inspiration from the behavior of real ant colonies.
  - Ants seem to be able to find their way, from nest to a food source and back, with relative easy.
  - Entomology studies discovered that in many cases this capacity is the result of interplay via chemical communication (Pheromone) between ants and the presence of many ants.
- Dorigo, Maniezzo & Colorni [1991]

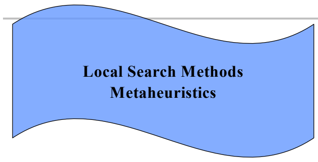
Ant Colonies

- Traveling Salesman Problem
  - Assign initially t_{ij} constant to each edge.
  - Each ant acts as follows:
    - At every step the ant chooses, using a probabilistic rule, a city to move among those not yet visited.
    - The probability of choosing edge (i,j) is directly proportional to the amount tij of pheromone in the edge and to inverse of edge length cij.
    - The ant remembers each city already visited.
    - After a tour, the ant lays a positive trail on each edge in the tour. And account some evaporation.
  - Now, consider M ants moving in the graph simultaneously. Ants laying trail on a set of edges make them more "desirable" for other ants.

Hybrid Metaheuristics

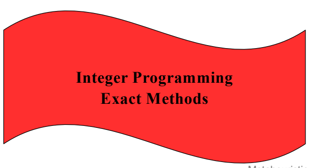
- HM combine various algorithmic components of different metaheuristics and even other optimization and artificial intelligent approaches.
  - …complementary metaheuristics, mathematical programming, simulation, constraint programming and machine learning…
- The main motivation behind the hybridization of different algorithms is to exploit the complementary character of different optimization strategies.
  - Blum, C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, *11*(6), 4135–4151.
  - Talbi, E. (2013), Hybrid Metaheuristics, Studies in Computational Intelligence, Springer.

Solution Methods

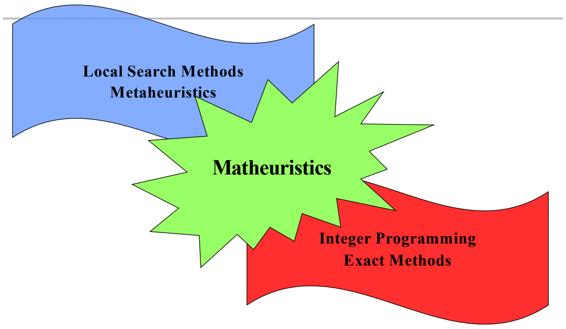


- Good solutions for complex and large-scale problems
- Short running times
- Easily adapted

- Mathematical proved optimal solutions
- Important information on the characteristics and properties of the problem.



Matheuristics



Matheuristics

- Refers to …
  - on exploiting mathematical programming (MP) techniques in a (meta)heuristic framework or
  - on granting to mathematical programming approaches the cross-problem robustness and constrained-CPU-time effectiveness which characterize metaheuristics.
    - Copied from **Matheuristics 2010 Conference webpage.**
- Integrative Combinations
  - Incorporating exact methods in metaheuritics
  - Incorporating metaheuristics in exact methods

Matheuristics: Classification

- Exact algorithms to explore large neighborhoods within local search.
- Information of high quality solutions found in several runs of local search is used to define smaller problems solvable by exact algorithms.
- Exploit lower bounds in constructive heuristics.
- Local search guided by information from integer programming relaxations.
- Use exact algorithms for specific procedures within metaheuristics.

Matheuristics

- Maybe the first application…
  - Use an exact algorithm to solve a sub-problem within a Local Search heuristic for the Job-Shop Scheduling Problem
    - **Perturbation Step: Solving to optimality the one-machine scheduling problem** with due dates and delivery times using the Carlier Algorithm.
    - **Iterated Local Search**
      - **Lourenço H.R.** (1995), Job-Shop Scheduling: computational study of local search and large-step optimization methods. European Journal of Operational Research **83**(2): 347-364. ISSN 0377-2217.
    - **Tabu Search**
      - **Lourenço HR.** and Zwijnenburg M. (1996), Combining the large-step optimization with tabu-search: application to the job-shop scheduling problem. In *Meta-Heuristics: Theory and Applications*, I.H. Osman and J.P. Kelly (Eds.), Kluwer Academic Publishers, pp. 219-236. ISBN: 978-0-7923-9700-7

Matheuristics: Examples

- Set Covering / Partitioning Problem
- Crew scheduling problem
  - Crossover operator considering the columns in the parents and **solving to optimality the reduced set covering problem**.
    - **Perfect Child /offspring**
  - Aggarwal, C.C., J.B. Orlin, and R.P. Tai. (1997). "An Optimized Crossover for Maximum Independent Set," Operations Research 45, 226-234.
  - **Lourenço H.R.**, Paixão J.P. and Portugal R. (2001), Multiobjective metaheuristics for the bus-driver scheduling problem. Transportation Science 35(3): 331-343.

Matheuristics: Examples

- Mixed Integer Programming
  - Construction of promising neighborhood using information contained in a continuous relaxation of the MIP model.
    - Relaxation Induced Neighborhood Search
  - Danna, E., Rothberg, E., & Pape, C. Le. (2005). Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming, Ser. A, 102, 71–90.
    - Network design and multicommodity routing
    - Job-shop scheduling with earliness and tardiness
    - Local branching

Matheuristics: Examples

- Vehicle Routing Problem
  - Iterated Local Search to assign customer to route and optimize the sequencing of the customers.
    - Solve a TSP using Concorde algorithm,
  - Dynamic programming is applied to determine the arriving time at each customer.
    - Ibaraki, Kubo, Masuda, Uno & Yagiura (2001)
  - Exploring a large scale neighborhood using dynamic programming in a VRP problem.
    - Thompson & Psaraftis(1993)

UNIVERSITAT POMPEU FABRA

Universitat Pompeu Fabra Barcelona

Matheuristics

- Real Applications
  - Maybe the best set of problems to apply Matheuristics methods…
  - Why?
    - Complex problems with a large number of constraints.
    - Sometimes difficult to model…
    - But, a simplification of the problem is frequently a well-studied optimization problem.
  - Apply metaheuristics for the real general problem, and exact methods for the well-known problem.

Metaheuristics 115

Universitat Pompeu Fabra Barcelona

Matheuristics

- Main References
  - Dumitrescu, I. and T. Stutzle (2003). Combinations of local search and exact algorithms. In G. R. Raidl ed. Applications of Evolutionary Computation. vol 2611 of LNCS, pp. 211-223. Springer.
  - Fernandes S. and **Lourenço H.R.** (2007), Hybrids Combining Local Search Heuristics with Exact Algorithms. In Proceeding of the V Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, MAEB'2007, F. Rodriguez, B. Mélian, J.A. Moreno, J.M. Moreno (Eds.) Tenerife, Spain, February 14-16, pp. 269-274. ISBN 978-84-690-3470-5.
  - Puchinger, J. and G. R. Raidl (2005). "Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification." Lecture Notes in Computer Science, vol. 3562.

Metaheuristics 116

Universitat Pompeu Fabra Barcelona

Simheuristics

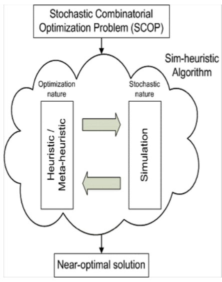
- Many real-world problems are NP-hard and actually also show **stochastic behavior**.
- If the system becomes complex, it's tempting to over-simplify the assumptions to have an analytically tractable model.
  - Advantage: you get exact (and deterministic) answers
  - but… **how good is such answer to a stochastic problem?**
- Simulation can deal with complex models...but…
  - Can get the best scenario?
  - How to deal with decision variable?
  - How to deal with optimization?

Metaheuristics 117

Universitat Pompeu Fabra Barcelona

Simheuristics

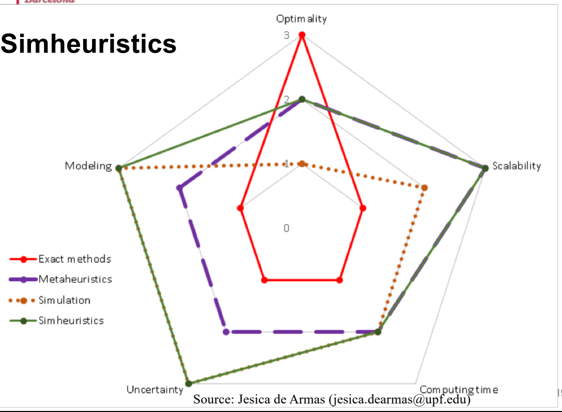
- A **Simheuristics** approach is a particular case of simulation-optimization (similar to simulation-based optimization, but focused on H/MH and SCOPs).
- Juan, A.A., Faulin, J., Grasman, S. E., Rabe, M., & Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, *2*, 62–72.
- Work of Juan, A.SA. And de Armas J.



Metaheuristics 118

Universitat Pompeu Fabra Barcelona

Simheuristics



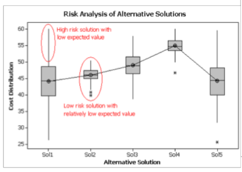
Source: Jesica de Armas (jesica.dearmas@upf.edu)

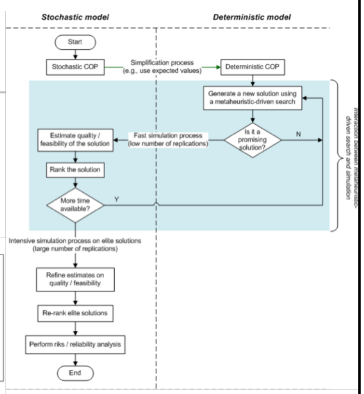
Metaheuristics 119

Universitat Pompeu Fabra Barcelona

Simheuristics

- Generate promising solution in the deterministic world.
- Validate these solution in the stochastic world.





Source: Jesica de Armas (jesica.dearmas@upf.edu)

Metaheuristics 120

UNIVERSITAT POMPEU FABRA

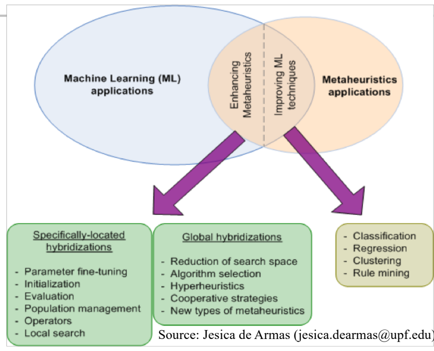
upf. Universitat Pompeu Fabra Barcelona

LearnHeuristics

- Inputs are deterministic (i.e., non-stochastic) but, **instead of being fixed in advance, they vary according to the structure of the solution,**
  - i.e., they change as the solution is being constructed following a heuristic-based iterative process.
- Existing literatura:
  - Machine learning is employed to enhance metaheuristics.
  - Metaheuristics are used to improve the performance of machine learning techniques.
- Calvet, L., Armas, J. De, Masip, D., & Juan, A. A. (2017). Learnheuristics: Hybridizing metaheuristics with machine learning for optimization with dynamic inputs. Open Mathematics, 15(1), 261–280.

upf. Universitat Pompeu Fabra Barcelona

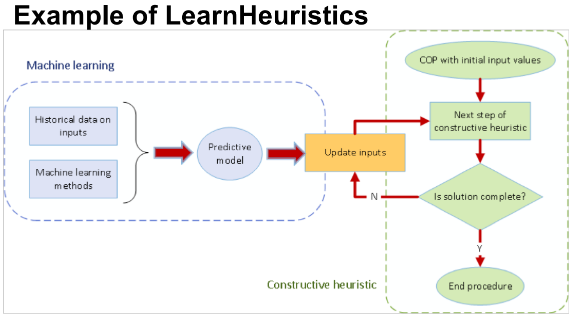
LearnHeuristics



Source: Jesica de Armas (jesica.dearmas@upf.edu)

upf. Universitat Pompeu Fabra Barcelona

Example of LearnHeuristics



Source: Jesica de Armas (jesica.dearmas@upf.edu)

upf. Universitat Pompeu Fabra Barcelona

Metaheuristics

- Which is the best metaheuristic?
  - begin with a simple method and then turn, if necessary, to a more complicated one or refine the first implementation
  - Small number of parameters
  - Evaluate its performance by:
    - Accuracy
    - Speed
    - Simplicity
    - Flexibility