METAHEURISTICS FOR HARD COMBINATORIAL OPTIMIZATION PROBLEMS

Éric Taillard

eivd*, Applied Univ. of Western Switzerland*

Wien, Nov. 2004 — Jan. 2005

SUMMARY OF THE LECTURE

1. Introduction (Tentative schedule : Mon. 10h-12h HS13)

Combinatorial optimization, complexity theory

Problem modelling

Heuristics and metaheuristics

2. Local searches (Tentative schedule :Mon. 14h-16h HS5, Tue. 15h-16h HS6)

Neighbourhood, move, improvement technique

Simulated annealing, threshold accepting, noising methods

Tabu search

3. Decomposition techniques (Tentative schedule :Wed. 14h-16h SR2)

POPMUSIC

4. Bio-inspired techniques (Tentative schedule :Tue. 16h-17h HS6, Thu. 10h-12h SR2)

Evolutionary algorithms, memetic algorithms

Scatter search, vocabulary building

Ant colonies

ORGANIZATION OF THE LECTURE

Lectures : 8.11 – 11.11

Presentations of research articles by students (1/2–3/4 hour, about 6-7 teams)

- Choice of an hard optimization problem, description of the problem
- State of the art, choice of an article
- Analysis of heuristic information exploited by the authors
- Analysis of meta-principles used by the authors
  - Problem modelling, neighbourhood, memory, population, …
- Description of the algorithm
- Discussion about numerical results, comments, …

Report : slides + copy of the article

Deadlines :

- By 12.11 2004 : choice of the subject
- 9.1–13.1 2005 Report + presentations

CHAPTER 1. INTRODUCTION

Example of a combinatorial optimization problem : the travelling salesman

Complexity theory

- Hard and easy problems
- *P, NP, NP-Complete, NP-Hard*

Problem modelling

- The quadratic assignment problem
- Football championship
- Turbine runner balancing

Heuristics and metaheuristics

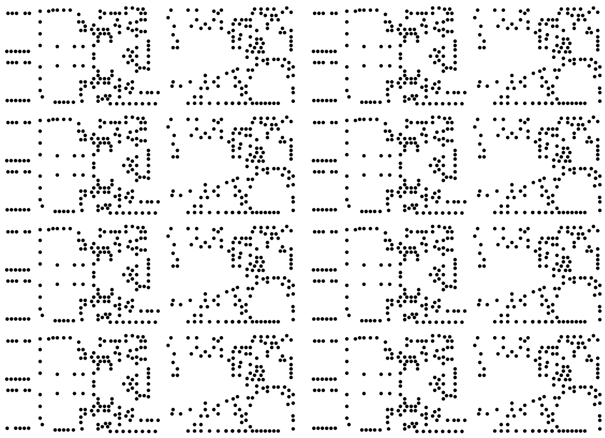
Short bibliography

EXAMPLE OF COMBINATORIAL OPTIMIZATION :

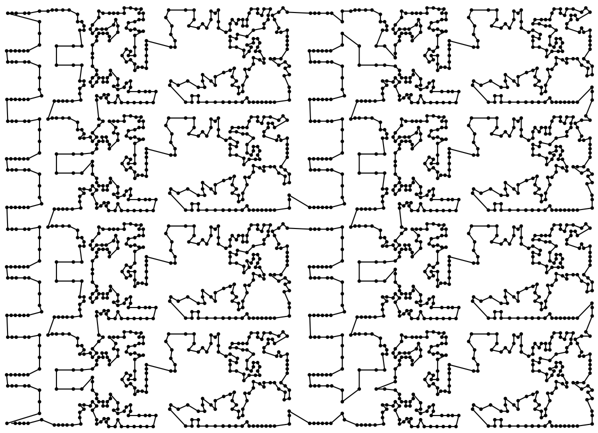
THE TRAVELLING SALESMAN PROBLEM

N cities, $\boldsymbol{D} = (d_{ij})$ distances matrix between cities i and j.

Problem : find the shortest tour passing exactly once in each city.



Problem instance



Optimal solution

HARD VERSUS EASY PROBLEMS

Complexity theory

M. R. Garey, D. S. Johnson , *Computers and Intractability: A guide to the theory of NP-completeness* Freeman, 1979

Very briefly :

$\pi \in P$ if π is a decision problem (answer = *yes* or *no*) and there is an **algorithm, polynomial** in the size of the data, for finding a solution (or proving that there is no solution) to the problem instance.

$\pi \in NP$ if there is a **polynomial algorithm** that can **check** if a given solution is correct.

π_1 can be **polynomially transformed** into π_2 if there is a function f that can be evaluated in polynomial time such that : s is a correct solution to $\pi_1 \Leftrightarrow f(s)$ is a correct solution to π_2

$\pi \in$ *NP-complete* if any problem of *NP* can be polynomially transformed into π.

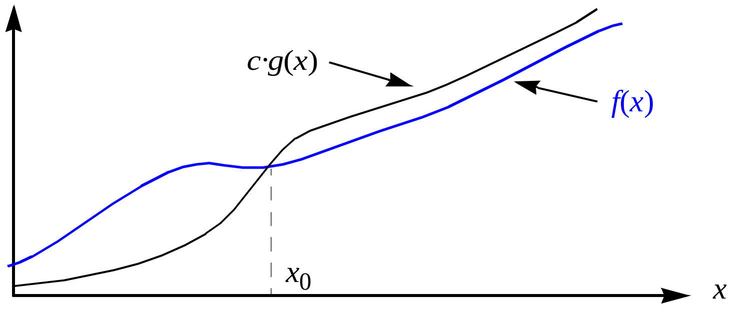
Class *NP-Hard* : extension of class NP-Complete to problems for which a solution is not belonging to (*yes*, *no*). Especially : optimization problems

ORDER OF A FUNCTION

Motivation : having an idea of the relative increase in computational effort when solving larger problem instances

Let f and g : 2 functions of a real variable x.

f is of order lower or equal to g if : $\exists\ x_0 > 0, c > 0$ such that $\forall\ x \geq x_0\ f(x) \leq c \cdot g(x)$



Notation $O(.)$

$f \in O(g)$

f in $O(g)$

Example : $f(x) = 1/x + \log(x) + 300x + 5x^2$ is in $O(x^2)$

EXAMPLE OF EASY AND HARD PROBLEMS

Easy problem : $\pi \in$ ***P***

Shortest path between vertices s and t in a network with N nodes.

Efficient algorithms finding the best solution are known.

Efficient = number of operations polynomial in problem size (Dijkstra $O(N^2)$, Bellman $O(N^3)$)

But : In general, there is an exponential number of paths between s and t

Hard problem : $\pi \in$ ***NP-Hard***

Travelling salesman problem.

No polynomial algorithm known.

It is conjectured that there are no.

Number of different solutions : $(N - 1)!$

PROBLEM MODELLING

Quadratic assignment problem

Football championship

Travelling salesman problem

Turbine runner balancing

THE QUADRATIC ASSIGNMENT PROBLEM (QAP)

Data : Matrices of flows and distances

Examples : Number of connections between electronic modules ;

Possible positions for the placement of the modules

Number of passengers that must change of aeroplane ;

Position of the gates in the airport

Frequency of consecutive appearance of two letters in a language ;

Time between typing 2 successive keys on a keyboard

Meeting frequency of two employees ;

Time for walking from one desk to another one

Objective : Assigning a position to each {module, aeroplane, key, employee} such that the total sum of products flows × distances is minimized

QAP : Mathematical model

Data : Flows matrix $\boldsymbol{F} = (f_{ij})$ $(i, j = 1, \ldots, n)$

Distances matrix $\boldsymbol{D} = (d_{rs})$ $(r, s = 1, \ldots, n)$

Solution : Permutation $\boldsymbol{p} = (p_1, p_2, \ldots, p_n)$ of n elements

Objective : Find a permutation $\boldsymbol{p}$ that minimizes : $\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot d_{p_i p_j}$

Example : f_{ij} = number of connections between modules i and j

d_{rs} = distance between locations r and s

QAP : example

Placement of 12 square electronic modules on a 3 × 4 rectangular board *a, b, …, l*

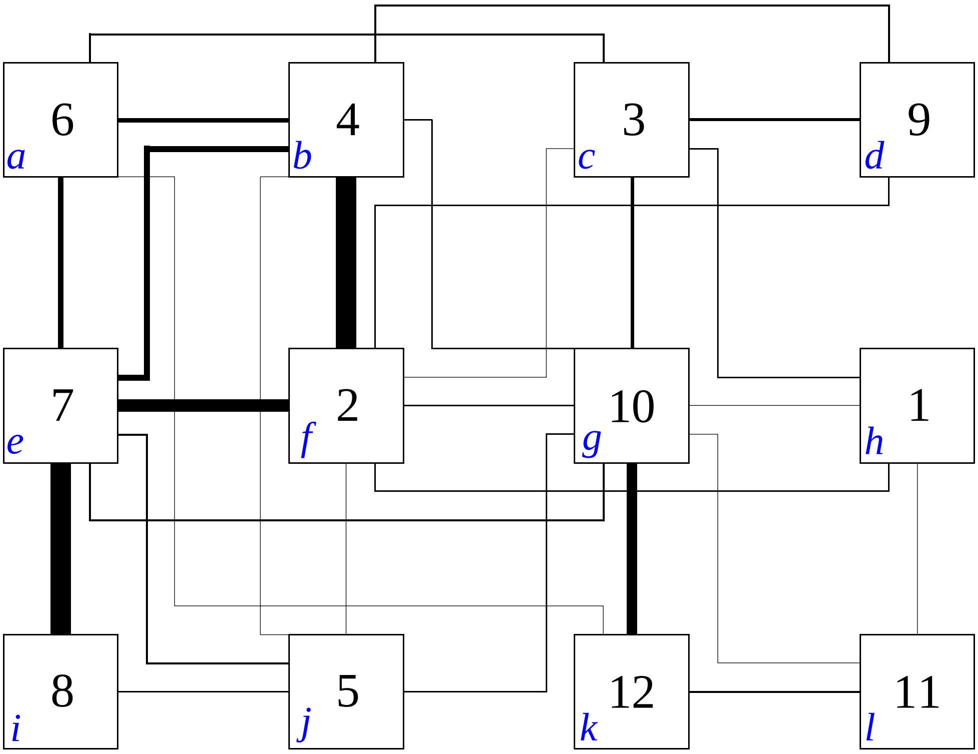
Connections can be placed either horizontally or vertically (Manhattan distances).

Flows matrix (number of connections)

	1	2	3	4	5	6	7	8	9	10	11	12
1	—	180	120	—	—	—	—	—	—	104	112	—
2	180	—	96	2445	78	—	1395	—	120	135	—	—
3	120	96	—	—	—	221	—	—	315	390	—	—
4	—	2445	—	—	108	570	750	—	234	—	—	140
5	—	78	—	108	—	—	225	135	—	156	—	—
6	—	—	221	570	—	—	615	—	—	—	—	45
7	—	1395	—	750	225	615	—	2400	—	187	—	—
8	—	—	—	—	135	—	2400	—	—	—	—	—
9	—	120	315	234	—	—	—	—	—	—	—	—
10	104	135	390	—	156	—	187	—	—	—	36	1200
11	112	—	—	—	—	—	—	—	—	36	—	225
12	—	—	—	140	—	45	—	—	—	1200	225	—

Number of connections between modules.

QAP : Optimal solution



Football championships

6 (generic) teams A, B, C, D, E, F wants to compete

Each team plays 2 times again each other team (home and away)

Matches are scheduled on 10 days, 3 matches simultaneously

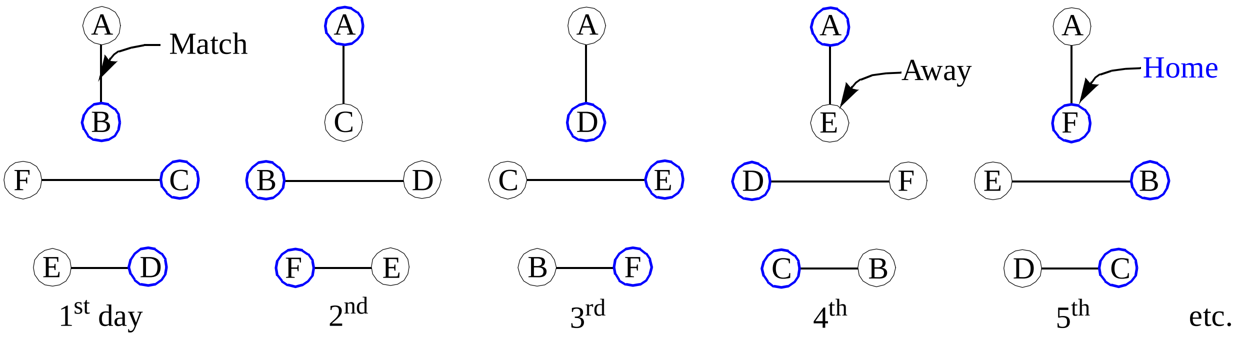
A team plays either home or away, at adverse team location

As many home and away matches for each team

No more than 2 successive away matches

Teams are travelling directly from the place of a match to the place of the next match.

Finding a solution respecting away-home constraints : Polygon rotation technique



TRIP PLANNING

Day	1st match	2nd match	3rd match
1	aB	Cf	De
2	Ac	Bd	eF
3	aD	bF	cE
4	Ae	bC	Df
5	aF	Be	Cd
6 = 1’	Ab	cF	dE
7 = 2’	aC	bD	Ef
8 = 3’	Ad	Bf	Ce
9 = 4’	aE	Bc	dF
10 = 5’	Af	bE	cD

Number of trips

	A	B	C	D	E	F
A	—	2	1	2	2	3
B	2	—		2	2	2
C	2	2	—		2	2
D	2	2	2	—		2
E	2	2	2	2	—	
F	2		3	2	2	—

Trips of team A : a → b → A → d → A → f → A → c → A → e → A

Trips of team B : B → B → f → c → B → a → d → B → B → e → b (no match)

Trips of team C : C → a → e → C → C → f → C → C → b → d → c

Trips of team D : D → b → D → D → c → e → D → a → f → D

Trips of team E : e → d → f → E → a → b → E → E → c → E → E

Trips of team F : f → c → F → F → d → F → F → e → b → F → a → f

MINIMIZING SUM OF TRAVEL COSTS

Assign an actual team (1, 2, 3, 4, 5, 6) to each generic team (A, B, C, D, E, F)

Number of trips

	A	B	C	D	E	F
A	—	2	1	2	2	3
B	2	—	0	2	2	2
C	2	2	—	0	2	2
D	2	2	2	—	0	2
E	2	2	2	2	—	0
F	2	0	3	2	2	—

Cost of the trip between cities

	1	2	3	4	5	6
1	—	34	1	12	72	53
2	34	—	2	43	36	5
3	1	2	—	21	52	33
4	12	43	21	—	42	37
5	72	36	52	42	—	12
6	53	5	33	37	12	—

→ Quadratic assignment problem.

Application to Brazilian national league football championship :

Spare 2’700’000$ of travel costs each year !

(with a more sophisticated method)

Travelling salesman problem

Data : Distance matrix $D = (d_{ij})$ between cities i and j

Objective : Find a permutation p minimizing :

$$\sum_{i=1}^{n-1} d_{p_i p_{i+1}} + d_{p_n p_1}$$

Let : $f_{ij} = \begin{cases} 1 \text{ if } j = i + 1 \text{ or } i = n \text{ and } j = 1 \\ 0 \text{ otherwise} \end{cases}$

Then :

$$\sum_{i=1}^{n-1} d_{p_i p_{i+1}} + d_{p_n p_1} = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot d_{p_i p_j}$$

→ **Quadratic assignment problem.**

But : More complex objective function evaluation ; there are very efficient neighbourhoods definitions for the TSP that are not easy to express as a modification of a permutation.

Turbine runner balancing :

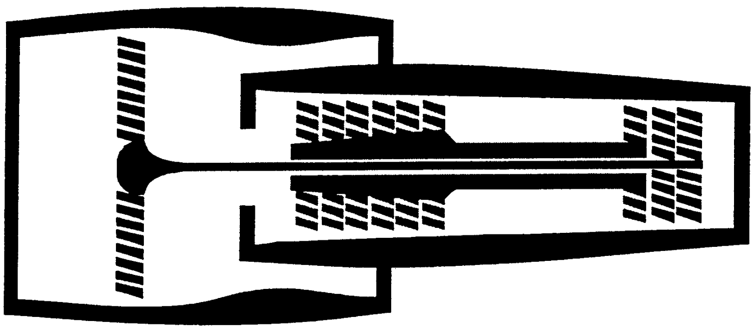
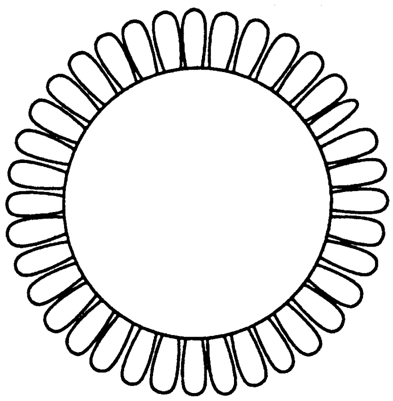


Fig. 1. Schematic figure of a jet engine.



Source : Mason & Rönnqvist, C&OR 24, 1997

n blades of weight w_i ($i = 1, \ldots, n$)

n angular positions $\theta_i = i/2\pi$ ($i = 1, \ldots, n$) or, more generally : Cartesian coordinates (x_i, y_i)

Objective : find a positions p_i ($i = 1, \ldots, n$) for each blade that minimizes : $\left(\sum_{i=1}^{n} w_i \cdot x_{p_i}\right)^2 + \left(\sum_{i=1}^{n} w_i \cdot y_{p_i}\right)^2$

TURBINE BALANCING

Alternate formulation (Laporte & Mercure, EJOR 35, 1988) :

Quadratic assignment problem with :

flows matrix $f_{ij} = w_i w_j$

distances matrix $d_{ij} = cos(\theta_i - \theta_j)$

Objective : find a permutation p that minimizes : $\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot d_{p_i p_j}$

Less general

Works only for angular positions

Container vessel loading ?

More complex

Objective computation $O(n^2)$ versus initial formulation $O(n)$

But : often used in the literature !

HEURISTIC

From greek : ηευρισκειν : to find

A heuristic method is based on **knowledge acquired by experience** on a given problem.

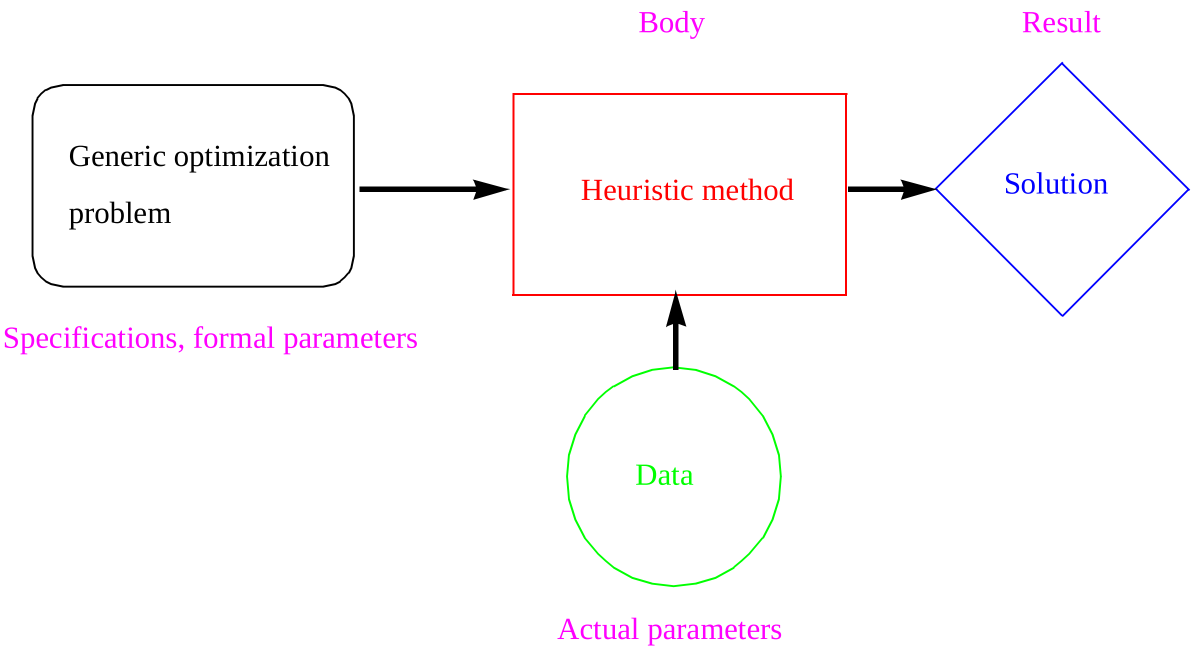
Heuristic methods (not necessarily providing the best possible solution) are opposed to :

Exact algorithms (that guarantee an optimal solution)

Example of a heuristic method for the travelling salesman problem :

```
Start form city 1
Repeat
   Go to the closest city not yet visited
Until all cities are visited
```

HEURISTIC METHOD : GENERAL FRAME

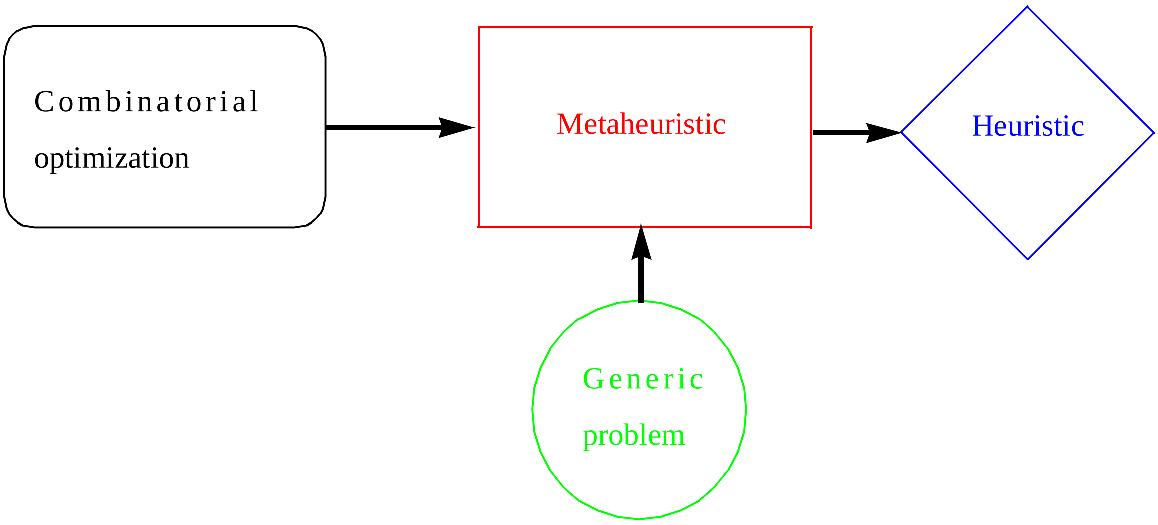


METAHEURISTIC

Meta- : from greek μετα that is above, that embeds

Metaheuristic : **Limited set of concepts**, that can be applied to a large set of combinatorial optimization problems and that allow to create **new heuristic methods**.

Support for designing heuristic methods, based on the knowledge acquired by designing heuristic methods for various problems.



EXAMPLES OF CONCEPTS

Problem modelling

Objective, constraints, feasible solutions

Neighbourhood structure

Small modification of a solution

Use of memory, use of random components, inspiration from physics or nature

Decomposition into sub-problems

Putting concepts together :

Simulated annealing

Tabu search

Evolutionary or genetic algorithms

Ants colonies

SHORT BIBLIOGRAPHY

J. Dréo, A. Pétrowski, P. Siarry, É. Taillard, *Métaheuristiques pour l'optimisation difficile*, Eyrolles, Paris, 2003. (translation in progress)

E. Aarts, J. K. Lenstra (éd.), *Local search in combinatorial optimization*, Princeton Univ. Press, Princeton, 2003 (2^{nd} edition).

T. Baeck, D. B. Fogel, Z. Michalewicz, *Evolutionary Computation*, Institute of Physics Publishing, 2000

D. Corne, M. Dorigo, F. Glover, *New ideas in optimization*, Mc Graw Hill, London 1999.

M. Dorigo, T. Stützle, *Ant Colony Optimization*, MIT Press, 2004.

F.Glover, M. Laguna, *Tabu Search*, Kluwer, Boston/Dordrecht/London, 1997.

Chapter 2. Local searches

General idea

Neighbourhood

Move

Solutions' space

Improvement technique

Neighbourhood choice, candidate list

Simulated annealing

threshold accepting, great deluge, noising methods

Tabu search

Tabu list

Long term memory

Advanced concepts : see also chapter 4

Local search

Observation

It is generally easy to find a (bad quality) solution to a combinatorial optimization problem

General idea

Generate an initial solution

Modify slightly the structure of this solution to get a better one

Repeat, until no improving modification is found

Modification of a solution : Neighbourhood

Example of neighbourhood for the QAP : transpositions (swaps)

1 2 3 4 5 6 9 7 8 ⟶ 1 2 7 4 5 6 9 3 8

Other possible neighbourhoods : swap adjacent elements, move an element at another location

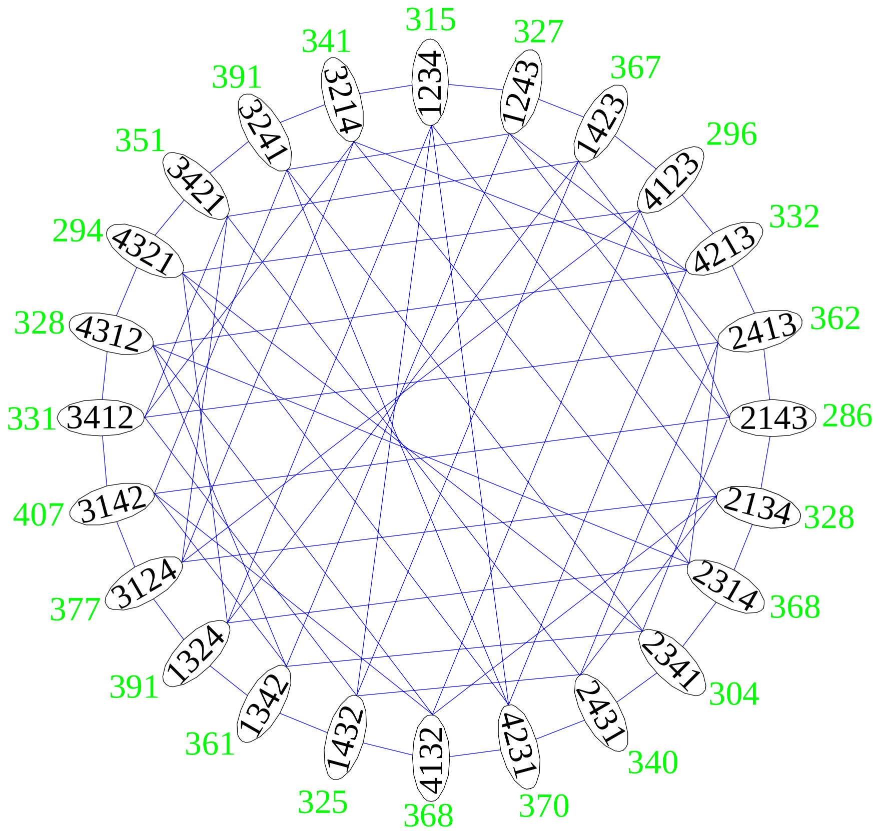
1 2 3 4 5 6 9 7 8 ⟶ 1 2 3 5 4 6 9 7 8

1 2 3 4 5 6 9 7 8 ⟶ 1 2 3 7 4 5 6 9 8

In general : For each solution $s \in S$ of the problem, define a set $N(s) \subset S$

Or : Define a set of move M that can be applied to any solution $s \in S$ i.e $N(s) = \{s' \mid s' = s \oplus m, m \in M\}$

Solutions space, neighbourhood structure

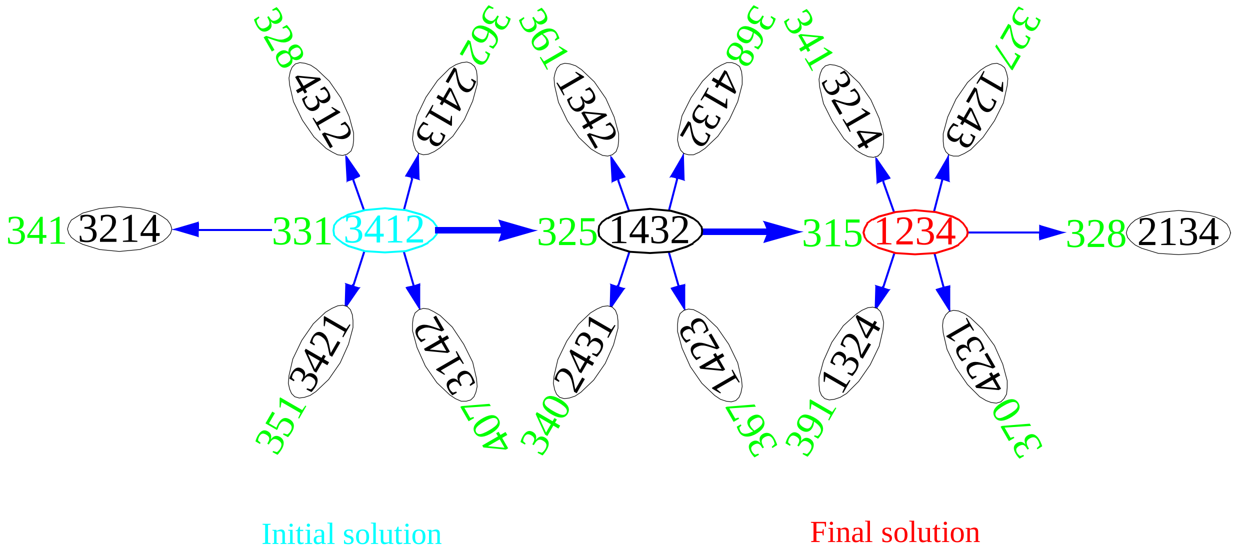


$$D = \begin{bmatrix} 0 & 6 & 2 & 6 \\ 6 & 0 & 7 & 5 \\ 2 & 7 & 0 & 9 \\ 6 & 5 & 9 & 0 \end{bmatrix}$$

$$F = \begin{bmatrix} 10 & 4 & 6 & 8 \\ 3 & 4 & 7 & 10 \\ 3 & 1 & 5 & 2 \\ 4 & 10 & 1 & 1 \end{bmatrix}$$

Space of the permutations of 4 elements with neighbourhood structure relative to transpositions. Value associated to each solution : objective value for a quadratic assignment problem instance.

DESCENT TOWARD A LOCAL OPTIMUM

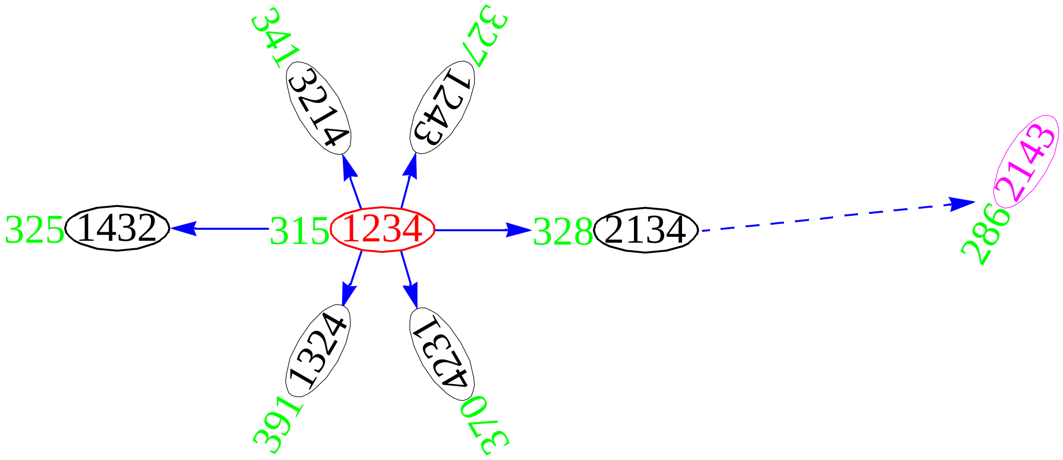


LOCAL SEARCH PSEUDO-CODE (MINIMIZATION, ∃ OTHER VARIANTS)

```
Data :          Initial solution s
                Objective function to minimize f
                Set M of moves that can be applied to any solution

Initializations :
   Continue := true

Main loop :
   Repeat
      Continue := false
      Best_neighbour_value := ∞
      For all m ∈ M, repeat
         If Best_neighbour_value > f(s ⊕ m) then
            Best_neighbour_value := f(s ⊕ m)
            Best_move := m
         End if
      End for
      If Best_neighbour_value < f(s) then
         Continue := true
         s := s ⊕ Best_move
      End if
   While Continue

Return : s -- Local optimum relatively to neighbourhood structure M
```

LOCAL OPTIMUM

The solution returned by a local search procedure is generally not the global optimum



The solution returned by a local search procedure is a local optimum (relatively to a given neighbourhood structure)

NEIGHBOURHOOD COMPUTATION

Problem : evaluate for solution *s* and move *m* $\Delta(s, m) = f(s \oplus m) - f(s)$

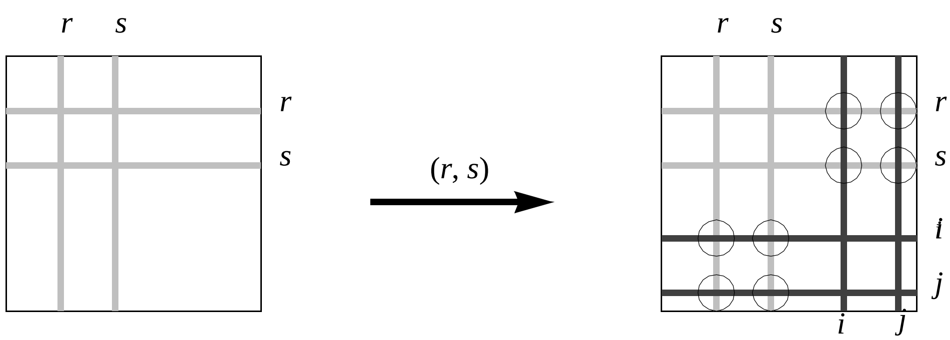
In case of the QAP :

$\boldsymbol{s} \equiv \boldsymbol{p}$ (= permutation) ; $m = (i, j)$ (= swapped elements)

Can be evaluated in $O(n)$:

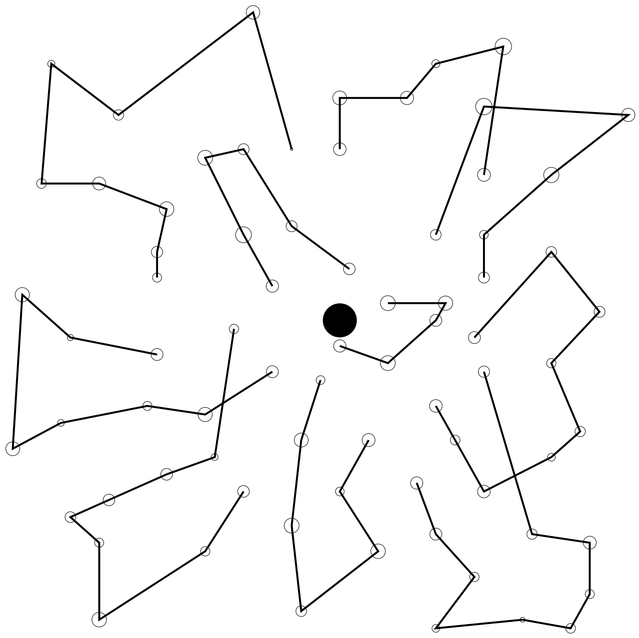
$$\Delta(\boldsymbol{p}, i, j) = \left\{ \begin{array}{l} (f_{ii} - f_{jj})(d_{p(j)p(j)} - d_{p(i)p(i)}) + (f_{ij} - f_{ji})(d_{p(j)p(i)} - d_{p(i)p(j)}) \\ + \sum\limits_{k \neq i, j} (f_{jk} - f_{ik})(d_{p(i)p(k)} - d_{p(j)p(k)}) + (f_{kj} - f_{ki})(d_{p(k)p(i)} - d_{p(k)p(j)}) \end{array} \right.$$

And even : can be evaluated in $O(1)$ in some cases



NEIGHBOURHOOD CHOICE

Definition of a good neighbourhood is a key element in many practical applications

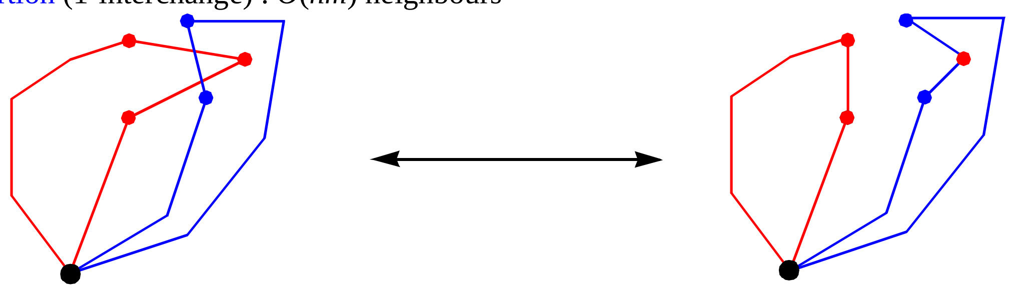


Solution to a vehicle routing problem : black disk = depot ; circles = customers, size = quantity

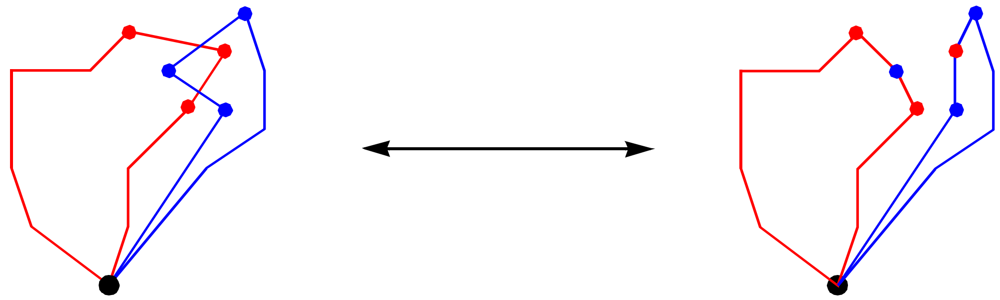
SIMPLE NEIGHBOURHOODS

Examples for the VRP (*n* customers, *m* tours)

Insertion (1-interchange) : $O(nm)$ neighbours



Exchange (2-interchange) : $O(n^2)$ neighbours

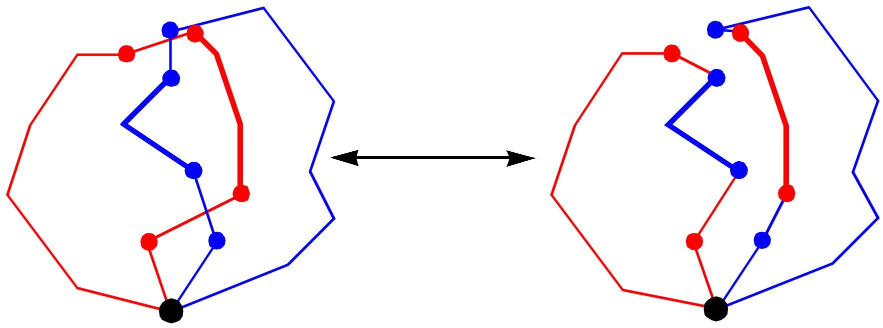


Generalization : λ-interchange : $O(n^\lambda)$ neighbours.

RESTRICTION OF NEIGHBOURHOOD SIZE

Candidate list (tabu search strategy)

Example 1 : CROSS-neighbourhood : $O(n^4/m^2)$ neighbours



Example 2 : Granular tabu search (Toth & Vigo 1998)

Consider only the shorter edges adjacent to each customers

$O(n^2) \rightarrow O(n)$

NEIGHBOURHOOD EXPANSION : COMPOSITE MOVES

Ejection Chains (tabu search strategy)

Avoid atomic changes

Expand neighbourhood size without increasing complexity too much

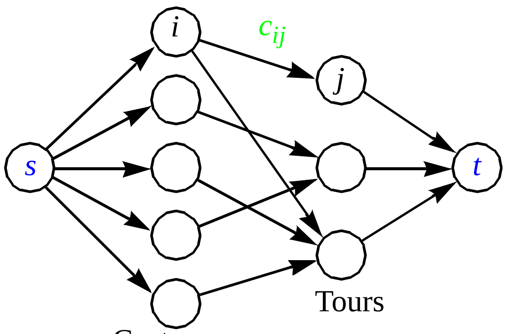
Perform jumps in solution space

Example for the VRP :

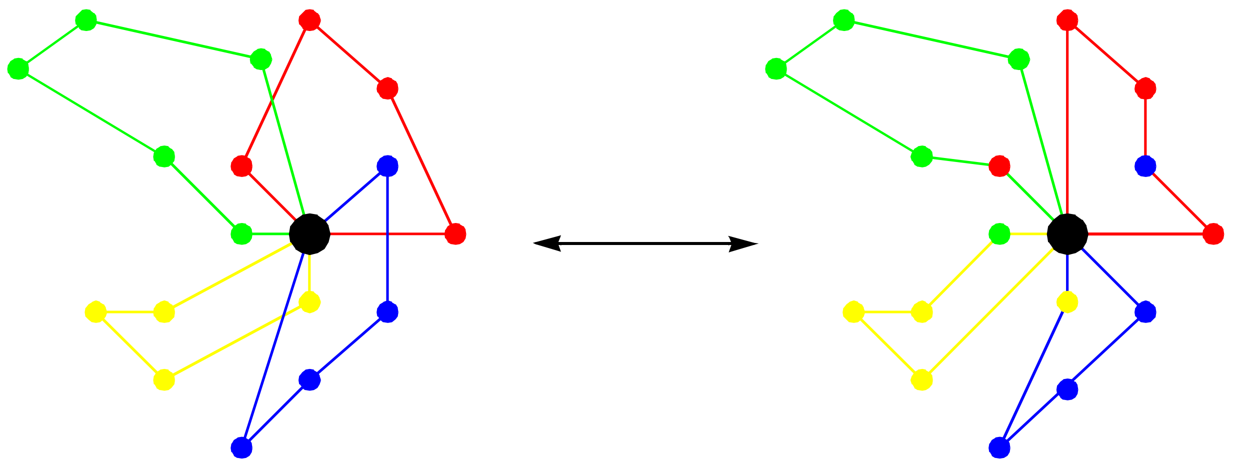
Network flow model (Xu & Kelly, Transp. Sci 30, 1996)

Find the max flow from s to t with lowest cost

c_{ij} : cost of removing customer i and inserting it in tour j

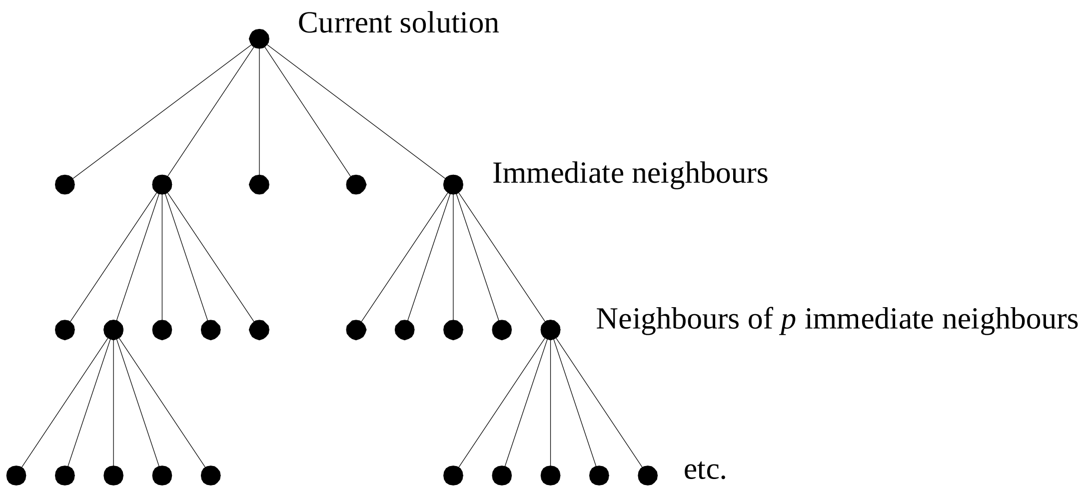


LARGE NEIGHBOURHOOD : “ ROTATION ”



Finding the best rotation with ejection chains : $O(nm^2)$

BEAM SEARCH

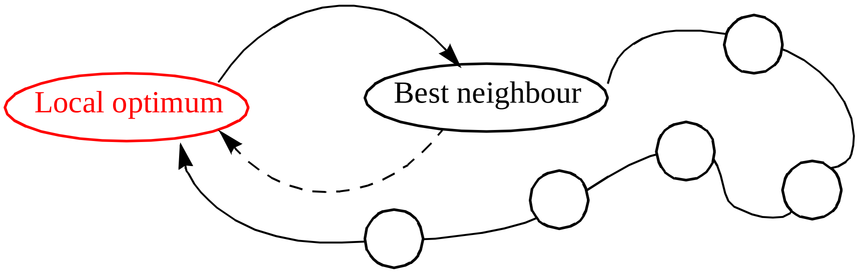


Simulated annealing

Observations :

The solution returned by an improvement technique can be far from global optimum

Continuing the search beyond a local optimum lead to cycling phenomenon



General idea :

Continue the search but :

Not deterministically

Improving moves are always accepted

The worse the move is, the lower the probability of choosing it is

Probability : in relation with particles energy changes in an annealing process

Simulated annealing : Pseudo-code

```
Data, Initialization :
              Initial solution s
              Objective function to minimize f
              Set M of moves that can be applied to any solution
              Initial temperature T_0, final temperature T_f
              Decreasing temperature factor 0 < α < 1
              Best solution found s* := s ; current temperature T := T_0

Main loop :
   Repeat
      Randomly choose a move m ∈ M
      Randomly generate u uniformly between 0 and 1
      Δ := f(s ⊕ m) – f(s)
      If e^(-Δ/T) > u then -- If Δ < 0 then m is always accepted
         s := s ⊕ m
         If f(s) < f(s*) then
            s := s*
         End if
      End if
      T := αT
   While T > T_f

Return : s*
```

Simulated annealing : practical tricks

Current temperature T is not diminished at each iteration. Examples :

Set $T := \alpha T$ after :

A given number of iterations (depending on problem and neighbourhood sizes e.g. : $100N$)

After a given number of improvement steps (e.g. $12N$)

Value of initial temperature T_0 :

Perform a given number of random moves, compute their average absolute value $\langle|\Delta|\rangle$

Choose an initial acceptance rate τ_0 of degrading moves (e.g. $\tau_0 = 30\%$)

Set $T_0 := -\langle|\Delta|\rangle/\ln(\tau_0)$ (solution of the equation $\tau_0 := e^{-\langle|\Delta|\rangle/T0}$)

Exiting the main loop :

Do not choose a stopping criterion depending on T_f but, e.g. :

Exit if s^* was not improved and temperature was decreased 3 times.

Temperature decreasing factor :

Value near to 1, e.g. $\alpha := 0{,}9$

Periodically re-heat the system ; in general : choose freely the sequence of temperatures

Simulated annealing : Convergence theorems

Simulated annealing process is considered as a Markov chain

Some results :

Under certain conditions (among them : infinite number of iterations), the simulated annealing process converges almost surely to the global optimum.

Under certain conditions (among them : the number of iteration larger than the square of total number of solutions), the stationary distribution of the Markov chain can be arbitrarily closely approximated

Practical implications ???

THRESHOLD ACCEPTING, GREAT DELUGE

Threshold accepting

Replace line :

```
If e^(-Δ/T) > u then -- If Δ < 0 then m is always accepted
```

by :

```
If Δ < T then
```

in simulated annealing code.

Interpretation of *T* :

Acceptable degrading threshold

Great deluge algorithm (maximizing) :

The line is replaced by :

```
If f(s ⊕ m) > T then
```

Interpretation of *T* :

Water level (absolute solution quality)

NOISING METHODS

Choose a sequence of distribution functions $Noise_k$ with (generally) :

standard deviation($Noise_{k+1}$) < *standard deviation*($Noise_k$)

At iteration k, add a random value distributed according to $Noise_k$:

Either to the problem data

Or to the move evaluation

Algorithm frame : similar to an improving method

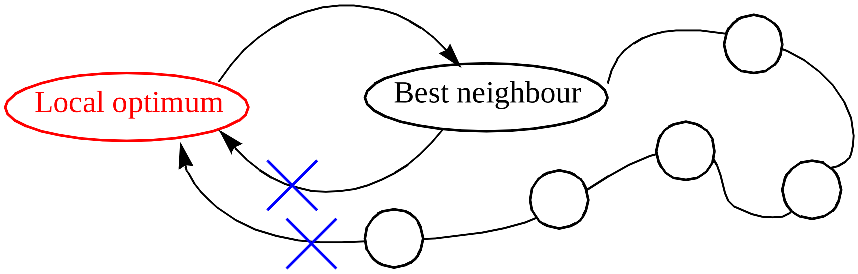
Property :

Noising methods are generalizations of simulated annealing, threshold accepting and great deluge algorithms

Tabu search

General ideas

Allow to search beyond first local optimum



Use memory instead of randomness for guiding the search

Prevent to come back to a solution already visited

Store visited solutions or

Store reverse of moves recently performed or

Store characteristics of solutions or moves recently performed

Solutions or moves stored are forbidden (**tabu**)

Basic Tabu search pseudo-code

```
Data, Initializations :
            Initial solution s
            Objective function to minimize f
            Set M of moves that can be applied to any feasible solution
            List (maximum of t elements) of tabu moves L = ∅
            Number I of iterations
            Best solution found s* := s
Main loop :
   For I iterations :
      Value_best_neighbour := ∞
      For all m ∈ M, m ∉ L repeat
         If Value_best_neighbour > f(s ⊕ m) then
            Value_best_neighbour := f(s ⊕ m)
            Best_move := m
         End if
      End for
      s := s ⊕ Best_move
      L := L ∪ Best_move^-1 -- The reverse of Best_move replace the oldest in L
      If f(s*) < f(s) then
         s* := s
      End if
   End for

Return : s*
```

Tabu search : Implementation tricks

Complete enumeration of the neighbourhood at each iterations

Try to find **algebraic simplification** for limiting the **complexity** of an iteration

If problem size and/or neighbourhood size are large

Store a limited set of best moves in a **candidate list**

For a number of iterations : evaluate only moves belonging to candidate list

Periodically update candidate list

Possible implementation of tabu list

Store the reverse of performed moves in a circular list of t elements (pseudo-code)

Effect : it is tabu to use the reverse move for t iterations

Store all enumerated solutions

Effect : it is tabu for t iterations to go back to a solution already visited

Difficulty : Memory requirement, complexity of solutions comparison

Practical implementation : Hashing function $h(s)$

Example of tabu list implementation for the QAP

Solution : permutation p of n elements

Move m that can be applied to any solution p : swap objects i and j ($m \equiv (i, j)$, $1 \le i < j \le n$)

i.e. object i, now in position p_i is moved on position p_j and object j on p_i

Reverse move definition :

Replace **simultaneously** object i on p_i **and** object j on p_j

Storing tabu conditions :

Matrix $\boldsymbol{T} = (t_{ir})$, of size $n \times n$

Meaning of t_{ir} :Iteration number after which it is again allowed to put object i on position r

$$(i, j) \text{ for permutation } p \text{ is tabu at iteration } k \Leftrightarrow (t_{ip_j} > k) \wedge (t_{jp_i} > k)$$

Tabu status of move (i, j) can be tested in constant time.

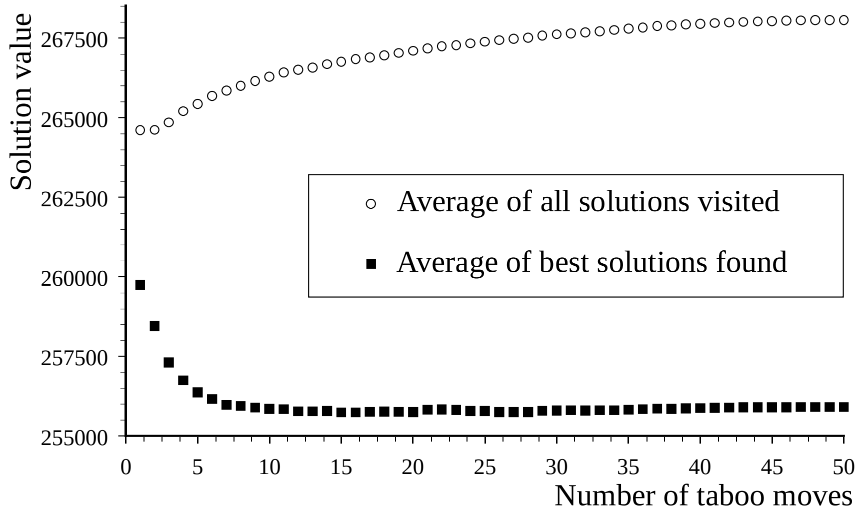
TABU DURATION (TABU LIST SIZE)

Small value of *t*

Cycling phenomenon may appear

Large value of *t*

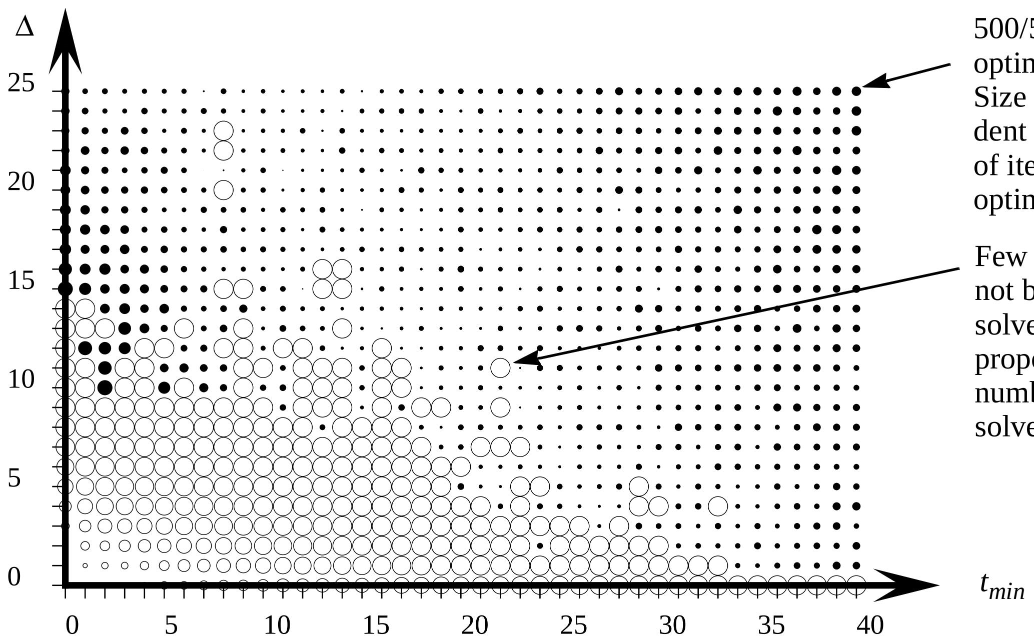
Too many forbidden moves implies visit of bad quality solutions



RANDOM TABU DURATION

Benefiting simultaneously from low and high tabu duration advantages

Randomly choose t between t_{min} and $t_{min} + \Delta$



500/500 problems optimally solved. Size of disk dependent of average number of iterations for finding optimum

Few instances have not been optimally solved. Circles size proportional to number of instances solved

LONG TERM MEMORY

Frequency penalty

Goal : Avoid to repeat a large number of time the same moves with small costs.

How :

For all move m, store its usage frequency $freq(m)$

Penalize of $F \cdot freq(m)$ the cost of move m

F : new parameter of the method

Forced moves

Goal : Break the structure of solutions visited by the search

How :

Identify moves never performed (e.g because they are too bad)

during the last K iterations (K : parameter).

Implementation for the QAP : can be immediately read in matrix T

TABU SEARCH : CONVERGENCE THEOREMS

Some results :

It is possible to define a probabilistic tabu search for which the hypothesis of convergence theorems of simulated annealing are satisfied.

A tabu search with exact tabu list (storing all visited solutions and choosing the oldest visited neighbour in case all neighbours are stored in the list) visits at least once the global optimum in a finite number of steps.

Chapter 3. POPMUSIC : Partial OPtimization Metaheuristic Under Special Intensification Conditions

Éric Taillard

EIVD

University of Applied Sciences of Western Switzerland

Yverdon-les-Bains, Switzerland

Content of the talk

Popmusic : general idea, frame, choices

Related concepts

Applications

- VRP
- Turbine runner balancing
- Clustering
- Cartographic labelling

Conclusions

POPMUSIC GENERAL IDEA

Start from an initial solution

Decompose solution into parts

Optimize a portion (several parts) of the solution

Repeat, until the optimized portions cover the entire solution

POPMUSIC FRAME

Solution $S = s_1 \cup s_2 \cup \ldots \cup s_p$ // p disjoint parts

$O = \varnothing$ // Set of " optimized " seed parts

While $O \neq S$, repeat // Improving method

1. Choose a seed part $s_i \notin O$
2. Create a sub-problem R composed of the r " closest " parts $\in S$ from s_i // r : parameter
3. Optimize sub-problem R
4. If R improved : set $O \leftarrow O \setminus R$

   Else : set $O \leftarrow O \cup s_i$

POPMUSIC CHOICES

Definition of a part

Distance between two parts

Parameter *r*

Optimization procedure

Variants :

slower :	set $O \leftarrow \varnothing$	instead of	set $O \leftarrow O \setminus R$
faster :	set $O \leftarrow O \cup R$	instead of	set $O \leftarrow O \cup s_i$

RELATED CONCEPTS

Candidate list, strongly determined and consistent variables (Glover)

“Chunking” (Woodruff)

Large neighbourhoods (Shaw)

VDNS (Hansen & Mladenovic)

Decomposition methods

POPMUSIC FOR VRP (TAILLARD 1993, ...)

Part:

Vehicle tour

Distance between parts:

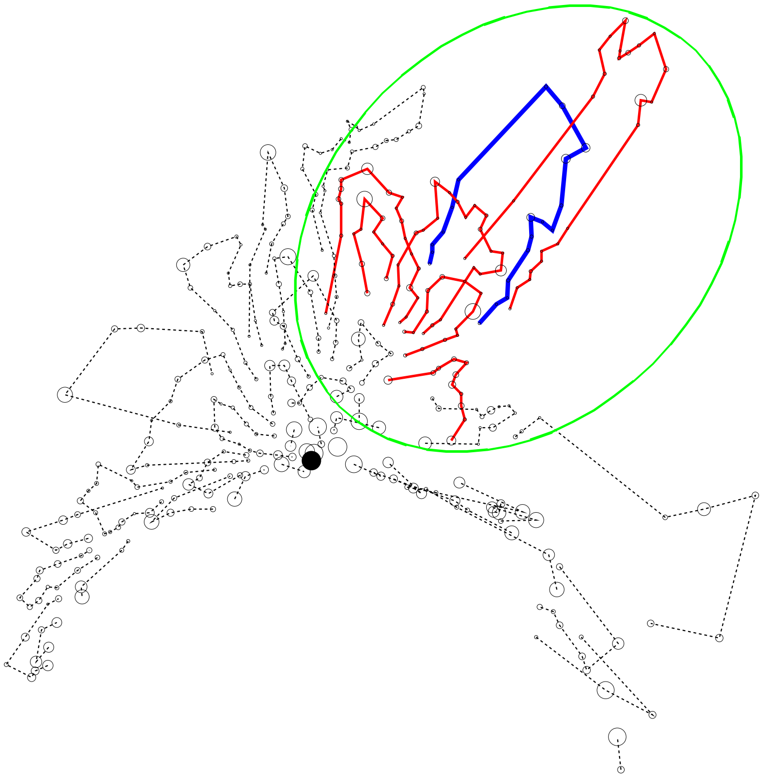
Polar distance between centres of gravity

Optimization process:

Basic tabu search

Particularity:

Many simultaneous optimization processes, treating all tours at each iterations



OTHER APPLICATIONS FOR VRP

Rochat & Semet 1994

Particularity: Repeat POPMUSIC with increased parameter *r*

First VNS application ?

Shaw 1998 (Large Neighbourhood)

Part: Customer

Distance: Euclidean distance + random component

Optimization process: Optimum or heuristic re-insertion (with constraint logic programming)

POPMUSIC FOR BALANCING PROBLEMS

Turbine runner balancing :

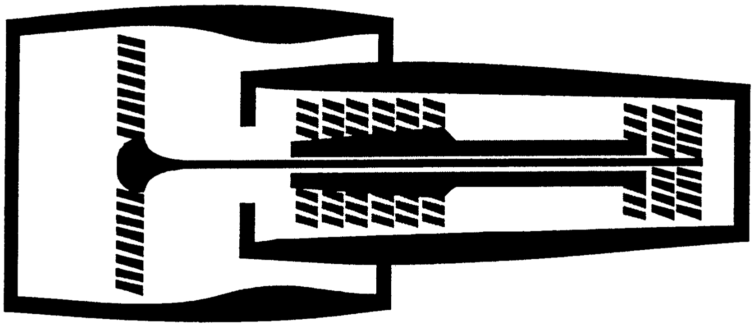
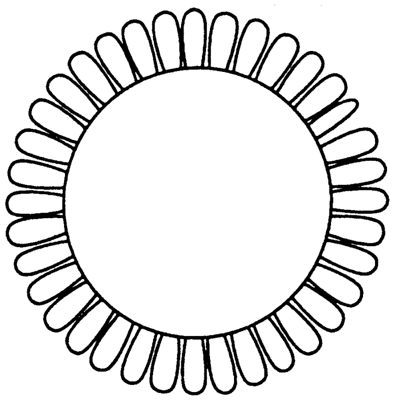


Fig. 1. Schematic figure of a jet engine.



Source : Mason & Rönnqvist, C&OR 24, 1997

n blades of weight w_i ($i = 1, \ldots, n$)

n angular positions $\theta_i = i/2\pi$ ($i = 1, \ldots, n$) or, more generally : Cartesian coordinates (x_i, y_i)

Objective : find a positions p_i ($i = 1, \ldots, n$) for each blade that minimizes : $\left(\sum_{i=1}^{n} w_i \cdot x_{p_i}\right)^2 + \left(\sum_{i=1}^{n} w_i \cdot y_{p_i}\right)^2$

TURBINE BALANCING

Alternate formulation (Laporte & Mercure, EJOR 35, 1988) :

Quadratic assignment problem with :

flows matrix $f_{ij} = w_i w_j$

distances matrix $d_{ij} = cos(\theta_i - \theta_j)$

Objective : find a permutation p that minimizes : $\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot d_{p_i p_j}$

Less general

Works only for angular positions

Container vessel loading ?

More complex

Objective computation $O(n^2)$ versus initial formulation $O(n)$

POPMUSIC CHOICES

Part:

Mechanical part

Distance:

Weight difference

Optimization process:

Basic taboo search (transposition neighbourhood, Taillard 1991)

NUMERICAL RESULTS (BASIC TABOO SEARCH)

Comparison with Reverse Elimination Method and Star Shape Diversification (Sondergeld & Voß 1996)

Turbines with 5..80 blades	REM/1	REM/10	SSA1/1	SSA1/10	SSA2/1	SSA2/10	Basic taboo search
$\sum$ excentricity	37.28685	37.55715	37.32012	37.47627	37.30784	37.30163	37.26166
$\sum$ CPU time [s]	> 5000	> 5000	> 5000	> 5000	> 11000	> 11000	109

NUMERICAL RESULTS

Number of blades	$100\left(\frac{\text{Basic taboo}(10000)}{\text{POPM}(22, 1000)} - 1\right)$	CPU POPM. [s. Sparc5]
30	7.2	3.5
40	40.0	7.2
50	52.1	12.8
60	49.4	18.0
70	76.0	21.4
80	65.2	26.5
100	23.3	37.2
100	56.0	55.5
100	35.3	65.1
100	56.6	49.3
100	11.5	58.5
100	32.0	46.8
100	77.7	57.9
100	47.1	51.3
100	51.3	62.5
100	26.2	45.1

POPMUSIC FOR CLUSTERING

Part :

Elements belonging to a cluster

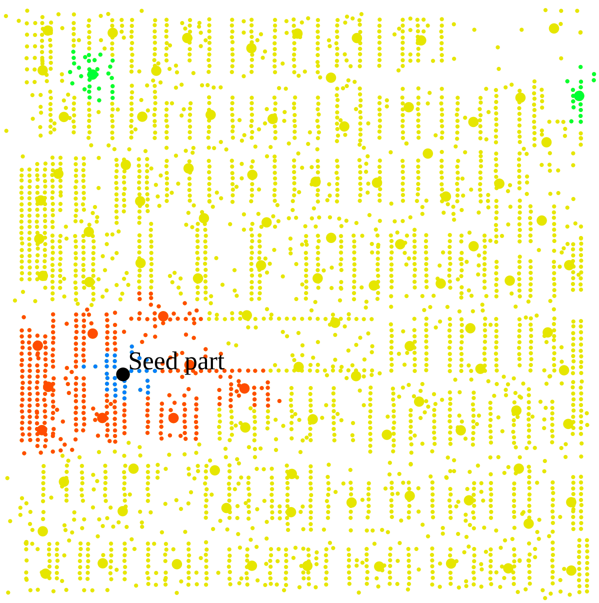
Distance :

Average dissimilarity between elements of different groups,

Distance between centres

Optimization process :

Improving method based on candidate list, relocation of a centre, stabilizing solutions (CLS)



NUMERICAL RESULTS

Minimum sum of squares

Problem instance : TSPLIB pcb3038.

Results of RVNS and VDNS from Hansen & Mladenovic 1999.

Problem		Quality [% above best known]				Computational time [s SPARC10]			
p	Best solution known	RVNS	VDNS	POPM. (6, 40)	POPM. (10, 100)	RVNS	VDNS	POPM. (6, 40)	POPM. (10, 100)
100	47685934.0	2.34	0.73	1.19	0.44	153	1132	145	505
150	30524769.8	3.13	1.44	1.16	0.58	153	1676	111	355
200	21875113.9	2.49	1.10	1.07	0.50	160	2124	96	262
250	16621446.4	2.56	1.34	1.35	0.76	182	2954	89	234
300	13289633.4	2.50	1.57	1.58	0.78	229	3151	82	205
350	11019171.4	2.60	1.36	1.69	0.76	231	3760	75	179
400	9362179.2	3.35	1.82	1.40	0.66	165	3446	72	170
450	8101618.7	3.47	1.71	1.61	0.80	242	4152	69	163
500	7102678.4	2.85	1.86	1.70	0.90	204	4060	68	156

NUMERICAL RESULTS

***p*-median problem instance TSPLIB RL11849**

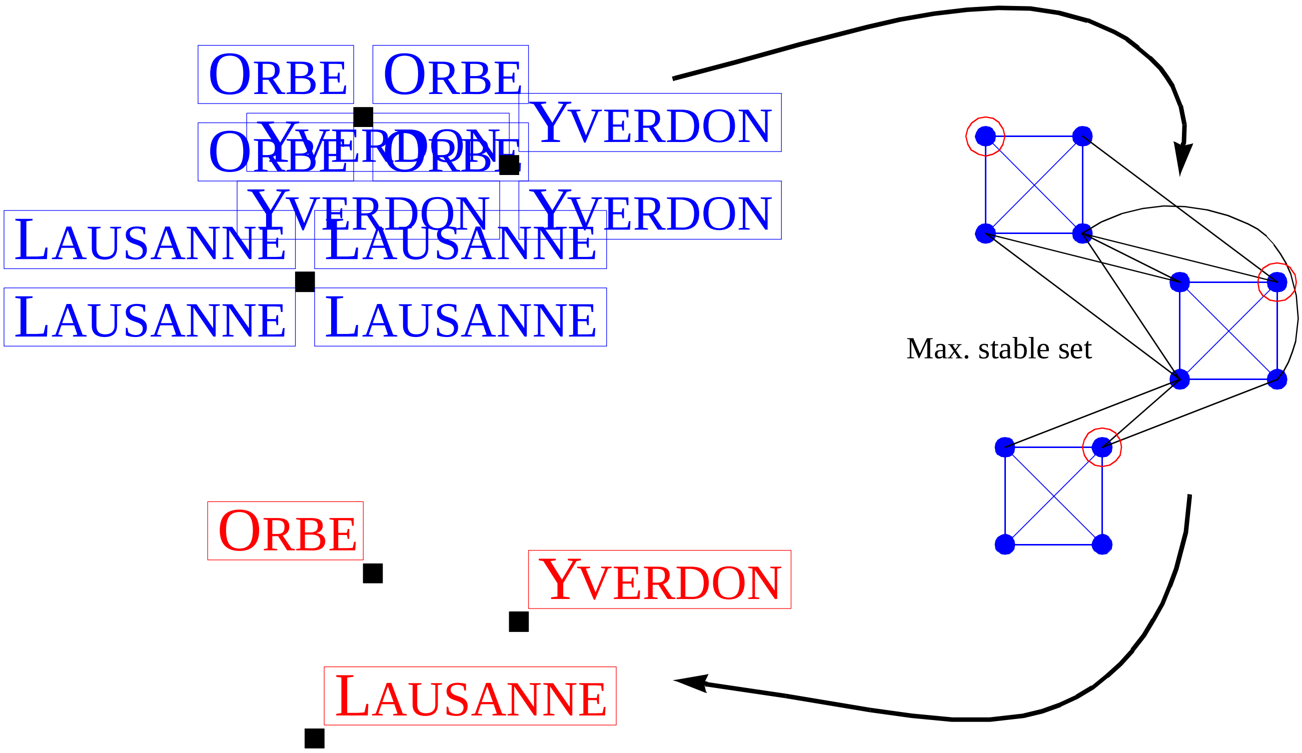
CGHM2004 : T.G. Crainic, M. Gendreau, P. Hansen, N. Mladenovic, " Cooperative Parallel Variable Neighborhood Search for the *p*-Median ", *Journal of Heuristics 10 (3)*, 2004, 293–314.

Equivalent run time CGHM2004 : 292000 seconds (= 80 hours, best) 7300 s. (average)

POPMUSIC implementation : Ph. Wälti, EIVD, collaboration Univ. Versailles (F).

p	r	Best published (CGHM 2004)	Quality (% above best published)			Time [s] (P.IV 1.8GHz)	
			CVVNS (1 proc.)	POPMUSIC (10)	POPMUSIC (100)	POPMUSIC (10)	POPMUSIC (100)
100	12	5855395.00	0.16	0.46	0.27	269	815
200	14	4017110.50	0.17	0.23	0.41	125	499
300	15	3210784.00	0.11	0.21	0.08	140	319
400	16	2712334.50	0.12	0.09	–0.05	142	277
500	16	2367523.00	0.12	0.21	0.14	150	205
600	17	2125355.50	0.12	0.04	–0.14	181	223
700	17	1932731.75	0.11	–0.01	–0.24	240	263
800	18	1775417.62	0.23	0.05	–0.14	299	286
900	18	1644025.75	0.18	–0.04	–0.19	328	306
1000	18	1531481.88	0.13	0.03	–0.31	376	323

CARTOGRAPHIC LABEL PLACEMENT



Other problem that can be modelized like this : assigning flight levels and departure times of aeroplanes.

POPMUSIC CHOICES

Part:

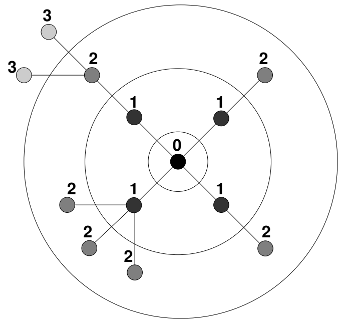
Object to label

Distance between parts:

Minimum number of edges needed to connect parts

Vertex ≡ object

Edge ∃ possible conflict in labelling the objects associated to vertices connected



Optimization process:

Tuned taboo search (Yamamoto, Camara, Nogueira Lorena, 2002)

Implementation : G. Burri, EIVD, 2003

NUMERICAL RESULTS

Method	Problem size				
% of good placements	100	250	500	750	1000
POPMUSIC	100	100	99.6	97.4	92.3
POPMUSIC fast	100	100	99.5	97.2	91.6
CGA (best) (Yamamoto, Nogueira Lorena, 2003)	100	100	99.6	97.1	90.7
CGA(av.) (Yamamoto, Nogueira Lorena, 2003)	100	100	99.6	96.8	90.4
Tabu (Yamamoto, Camara, Nogueira Lorena, 2002)	100	100	99.2	96.8	90.00
GA with masking (Verner, Wainwritht, Schönenfeld, 1997)	100	99.98	98.79	95.99	88.96
GA (Verner, Wainwritht, Schönenfeld, 1997)	100	98.40	92.59	82.38	65.70
Simulated Annealing (from Christensen et al. 1995)	100	99.90	98.30	92.30	82.09
Zoraster(from Christensen et al. 1995)	100	99.79	96.21	79.78	53.06
Hirsh (from Christensen et al. 1995)	100	99.58	95.70	82.04	60.24
3-Opt Gradient Descent (from Christensen et al. 1995)	100	99.76	97.34	89.44	77.83
2-Opt Gradient Descent (from Christensen et al. 1995)	100	99.36	95.62	85.60	73.37
Gradient Descent (from Christensen et al. 1995)	98.64	95.47	86.46	72.40	58.29
Greedy (from Christensen et al. 1995)	95.12	88.82	75.15	58.57	43.41
CPU time (Pentium III, 745MHz, ?)					
POPMUSIC	0.0	0.0	0.3	3.5	20
POPMUSIC2 fast	0.0	0.0	0.2	1.3	4
CGA (best) (Yamamoto, Nogueira Lorena, 2003)	0	0.6	21.5	228	1227
CGA(av.) (Yamamoto, Nogueira Lorena, 2003)	0	0.6	21.5	196	982
Tabu (Yamamoto, Camara, Nogueira Lorena, 2002)	0	0	1.3	76	353

CONCLUSIONS

Easy to implement

Simple frame

Basic improvement method (no candidate list, intensification, diversification)

Can be used with an exact method

Low complexity

Grows typically linearly with problem size

Future

Better implementations for *p*-median, map labelling, VRP

Projects and collaborations with other teams ?

Chapter 4. Bio-inspired techniques

Placement of emitter antennas

Evolutionary (genetic) algorithms

Hybrid evolutionary algorithms (memetic algorithms)

Generalization, but not bio-inspired

- Scatter search
- Vocabulary building

Ant systems

Placement of emitter antennas

Data

m potential positions

- Building cost of an antenna in position j : c_j ($j = 1, \ldots, m$)
- Maximal capacity of antenna placed in position j : k_j ($j = 1, \ldots, m$)

n customers

- Capacity requirement of customer i : d_i ($i = 1, \ldots, n$)
- Cost for customer i if it is connected to antenna in position j :

  s_{ij} ($i = 1, \ldots, n, j = 1, \ldots, m$)

Decision variables

$y_j = 1$: An antenna is buil in position j ($j = 1, \ldots, m$)

(= 0 otherwise)

$x_{ij} = 1$: Customer i is connected to antenna j ($i = 1, \ldots, n, j = 1, \ldots, m$)

(= 0 otherwise)

Mathematical formulation

minimize	$\sum_{j=1}^{m} c_j y_j + \sum_{i=1}^{n} \sum_{j=1}^{m} s_{ij} x_{ij}$		objective
under	$\sum_{j=1}^{m} x_{ij} = 1$	$\forall i$	All customers are connected
contraints	$\sum_{i=1}^{n} d_i x_{ij} \le k_j y_j$	$\forall j$	Antennas must not be saturated
	$0 \le x_{ij} \le 1$ integers	$\forall i, j$	
	$0 \le y_j \le 1$ integers	$\forall j$	

Simplification : capacity constraints are omitted

$$x_{ij} \le y_j, \forall i, j$$

Evolutionary Algorithms

Darwin (1859)

Evolution of living species is based on competition that selects the more adapted individuals, insuring them to have many children, thus a transmission of useful characteristics.

Neo-Darwinism :

Genetic mutations must be taken into consideration

In the 50's, this process was simulated to solve problem from industrial engineering.

Emergent approaches :

Evolutionary strategies (Schwefel et Rechenberg, 1965)

Optimization of continuous parameters

Evolutionary programming (Fogel Owens, Walsh, 1966)

Artificial intelligence alternative for modelling the evolution of finite states automata

Genetic algorithms (Holland, 1975)

Understanding self-adaptation mechanisms

NATURAL EVOLUTION PROCESS

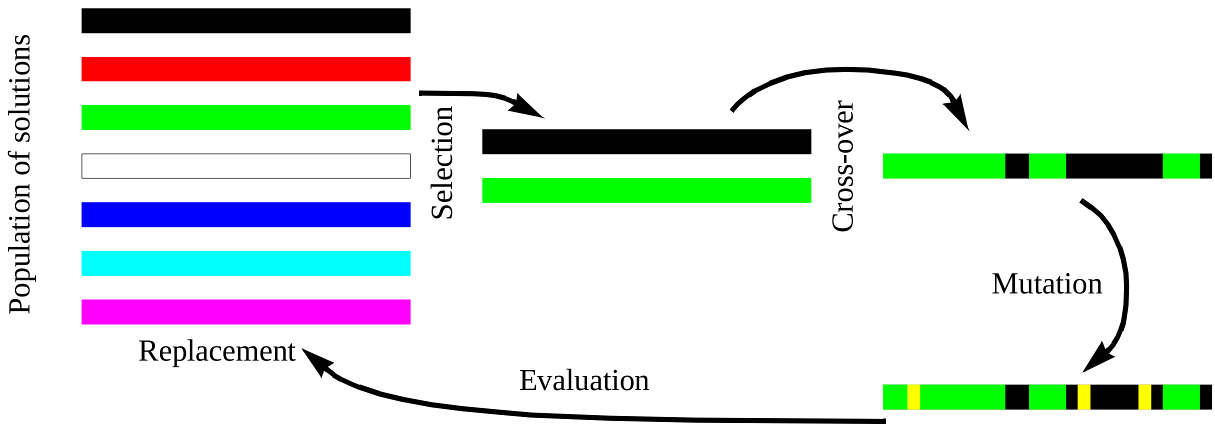
Selection of 2 individuals from a population.

Cross-over of these 2 individuals, in a random fashion

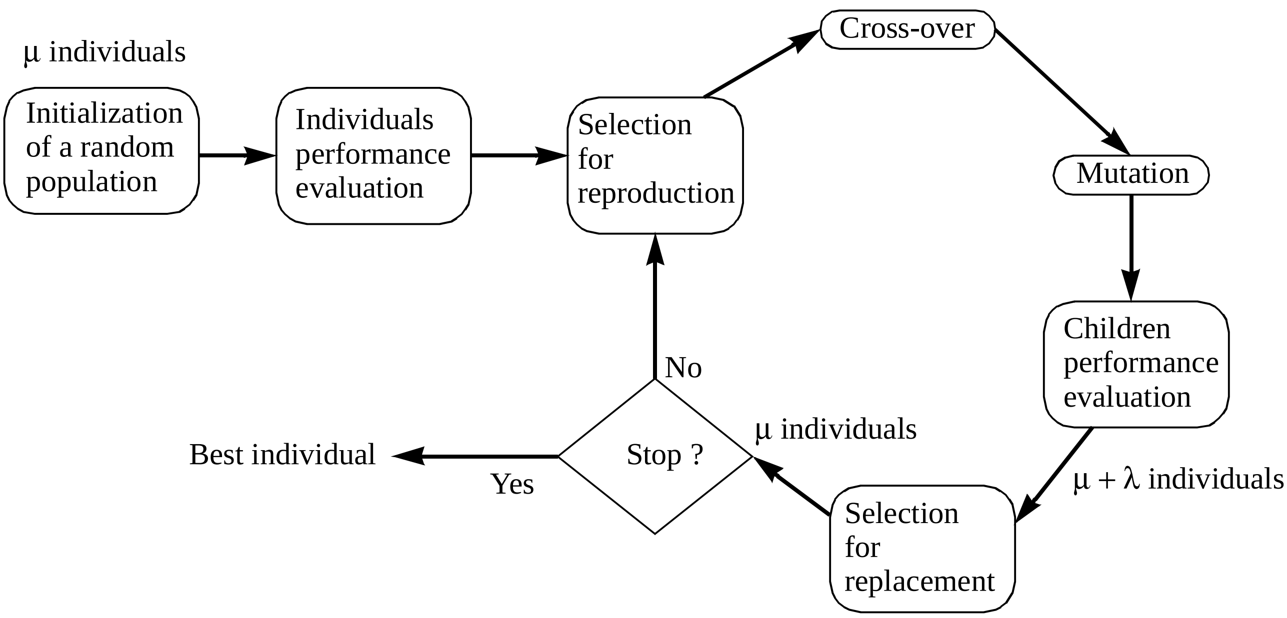
Mutations to create a new individual, in a random fashion.

Evaluation of the new individual, entering the population.

Replacement of individuals that are too weak, old, ill



WORKING SCHEME



EXAMPLE OF RUN OF AN EVOLUTIONARY ALGORITHM

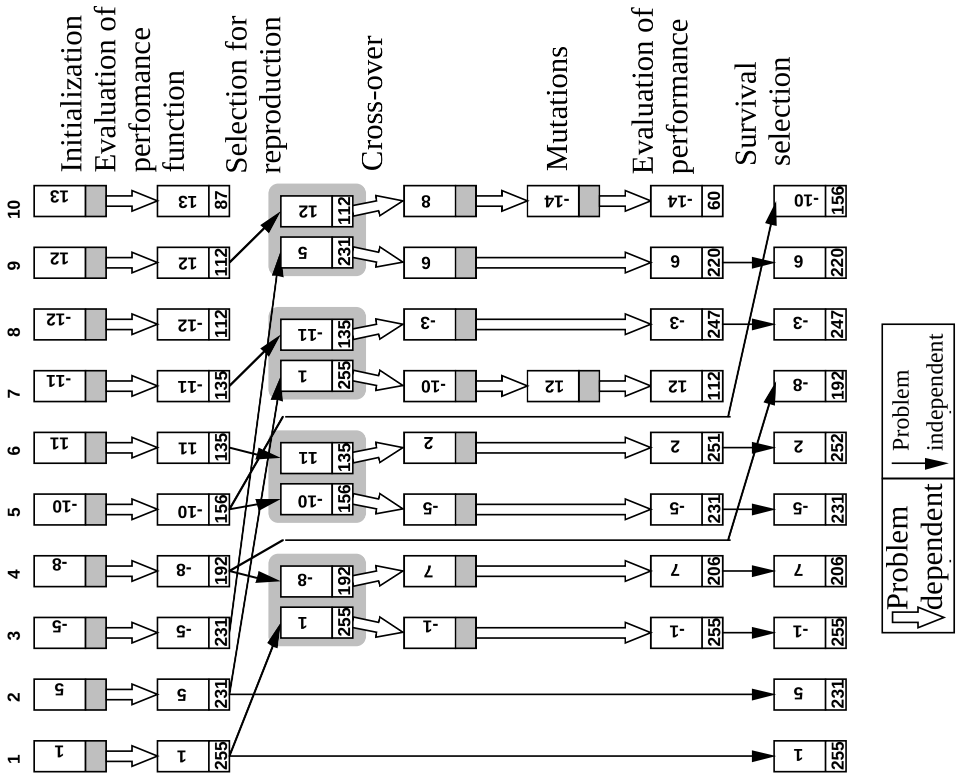
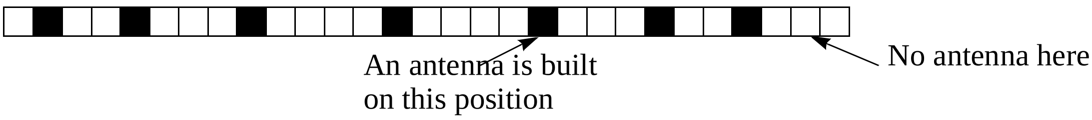
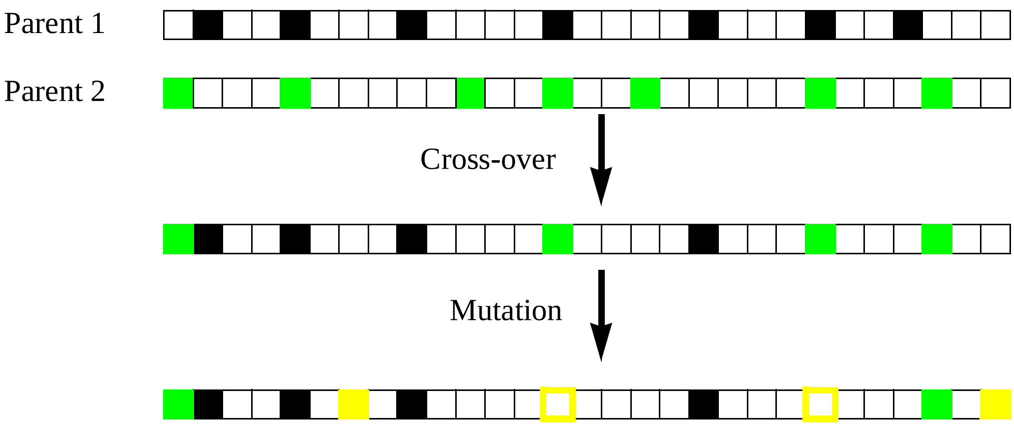


ILLUSTRATION IN CASE OF ANTENNA PLACEMENT :

Coding a solution

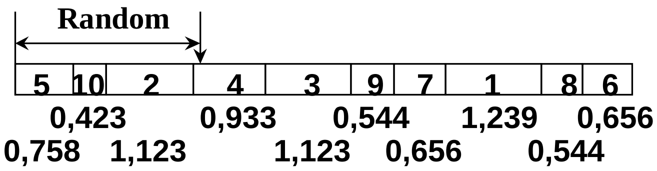




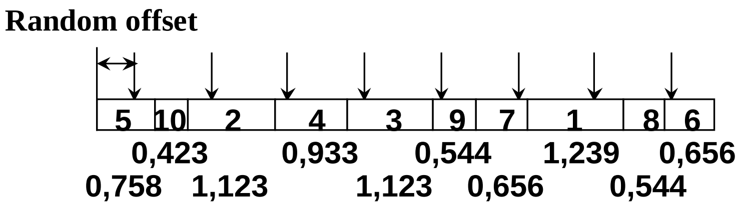
Simulation of reproduction process

PROPORTIONAL SELECTION FOR REPRODUCTION OPERATORS

Roulette wheel



Stochastic universal sampling



Rank

$$r = \mu \cdot (1 - \sqrt[p]{U(0, 1)})$$

$p = 1$: uniform random selection ; $p > 1$: best individuals favoured

SELECTION FOR REPLACEMENT OPERATOR

Generations

Only children are selected from one population to the next one (parents are removed)

($\lambda = \mu$)

(μ, λ)-Evolutionary Strategy

Only the μ best children are kept ($\lambda > \mu$)

Stationary

Replace λ parents by children (λ small, typically 1 or 2)

Elitist

From the complete population of $\mu + \lambda$ individuals, keep only the μ best

Cross-over operators

1-point

Parents — 0–1 vectors — Children

Point 1

1	0	1	1	0	1	0	1		1	0	1	0	1	1	1	0
								→								
0	0	1	0	1	1	1	0		0	0	1	1	0	1	0	1

2-points

Parents — 0–1 vectors — Children

Point 1 Point 2

1	0	1	1	0	1	0	1		1	0	1	0	1	1	0	1
								→								
0	0	1	0	1	1	1	0		0	0	1	1	0	1	1	0

Permutations : OX

1	3	8	2	7	4	5	6		8	2	4	7	6	1	5	3
								→								
8	3	4	7	6	1	5	2		3	6	1	2	7	4	5	8

Uniforms cross-over

Parents — 0–1 vectors — Children

Point 1 Point 2

1	0	1	1	0	1	0	1		1	0	1	0	0	1	1	1
	↕		↕			↕		→								
0	0	1	0	1	1	1	0		0	0	1	1	1	1	0	0

Parents — Permutation — First child

1	3	8	2	7	4	5	6	⇒	1	3	4	2	6		5	
												⇓				
8	3	4	7	6	1	5	2		1	3	4	2	6	7	5	8

MUTATION OPERATORS

0 – 1 vectors :

Modify each bit randomly, with a given probability

Modify a given number of bits randomly

Other cases :

cf. concept of *move* in local search

GENETIC DRIFT

The repetition of the loop selection — cross-over — replacement, tends to make the population homogeneous.

Finally, the population contains multiple identical individuals.

Slowing down the convergence speed

Increase population size

Lower selection pressure (for reproduction and replacement)

Increase mutation rate

CONVERGENCE OF EVOLUTIONARY ALGORITHMS

There are convergence (global optimum found) theorems.

Among assumptions :	Mutation probability not 0
Consequence :	The probability of generating the optimum is not 0

Not really stronger than random solutions generation !

HYBRID EVOLUTIONARY ALGORITHMS

Weaknesses of standard evolutionary algorithms :

Solution coding (Boolean vectors)

Population convergence (genetic drift)

Solutions not locally optimal

Strength of evolutionary algorithms

Good cover of solutions' space

Weaknesses of local searches

Bad cover of solution's space

Simultaneously benefiting from evolutionary algorithms and local searches advantages :

Replace mutation operator by a local search (improving method, simulated annealing, tabu search, …), specific population management, specific cross-over operators

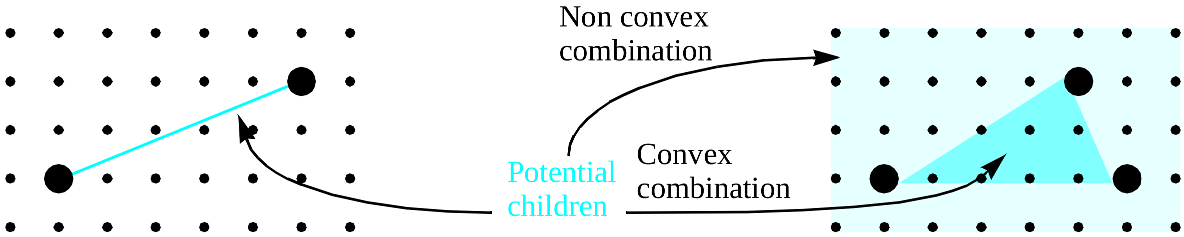
→ Memetic algorithms, scatter search, vocabulary building.

SCATTER SEARCH

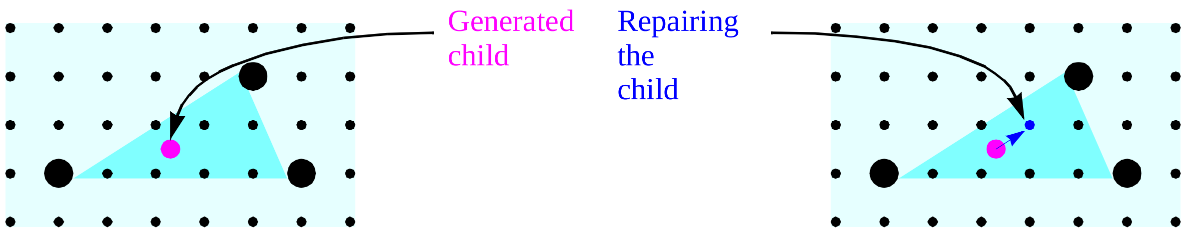
First approach : Glover 1977 (Integer linear programming)

Replace binary coding scheme by a natural scheme (vector of integers)

Replace cross-over operator by a linear combination of 2 or more vectors

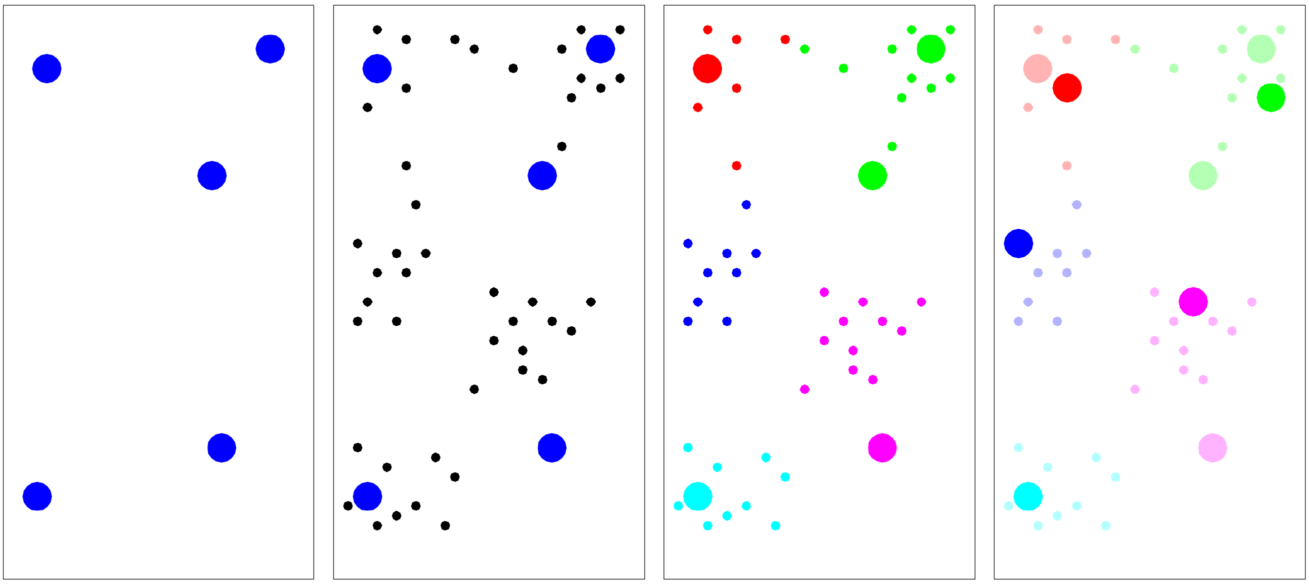


Replace mutation operator by a repair/projection operator



Intelligent population management (elite solutions)

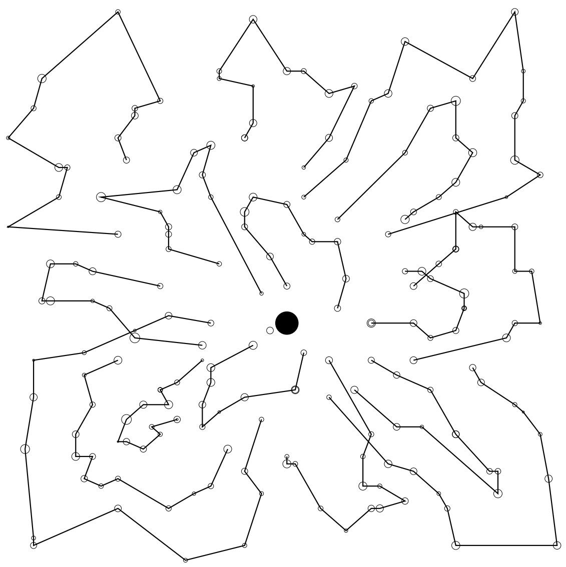
POPULATION MANAGEMENT



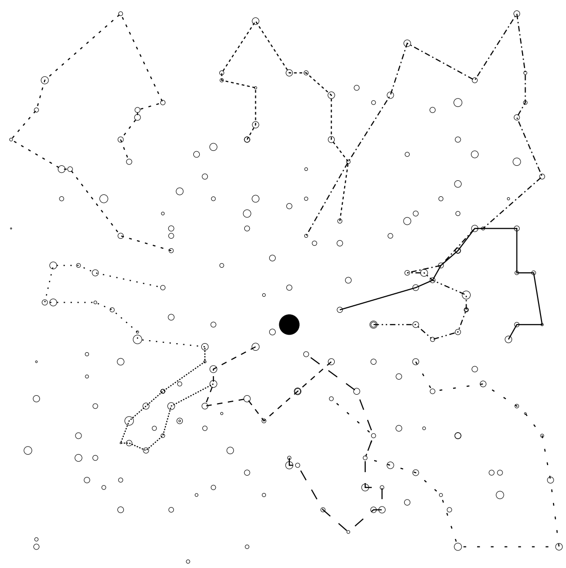
Initial population (elite solutions) | Evolution | Clustering | New elite solutions (the best in each cluster)

Remark : Clusters can be found by solving an antenna positioning problem

VOCABULARY BUILDING



An extremely good solution to a VRP instance



Few tours found after few tabu search iterations

General idea :

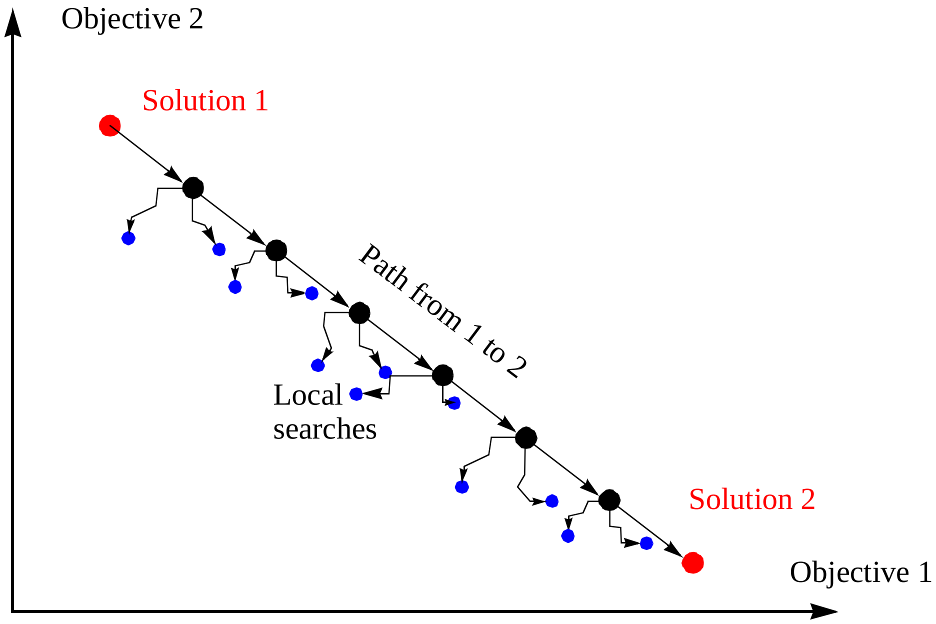
Store a population of parts of solutions (= words)

Build new solutions (= sentences) with words

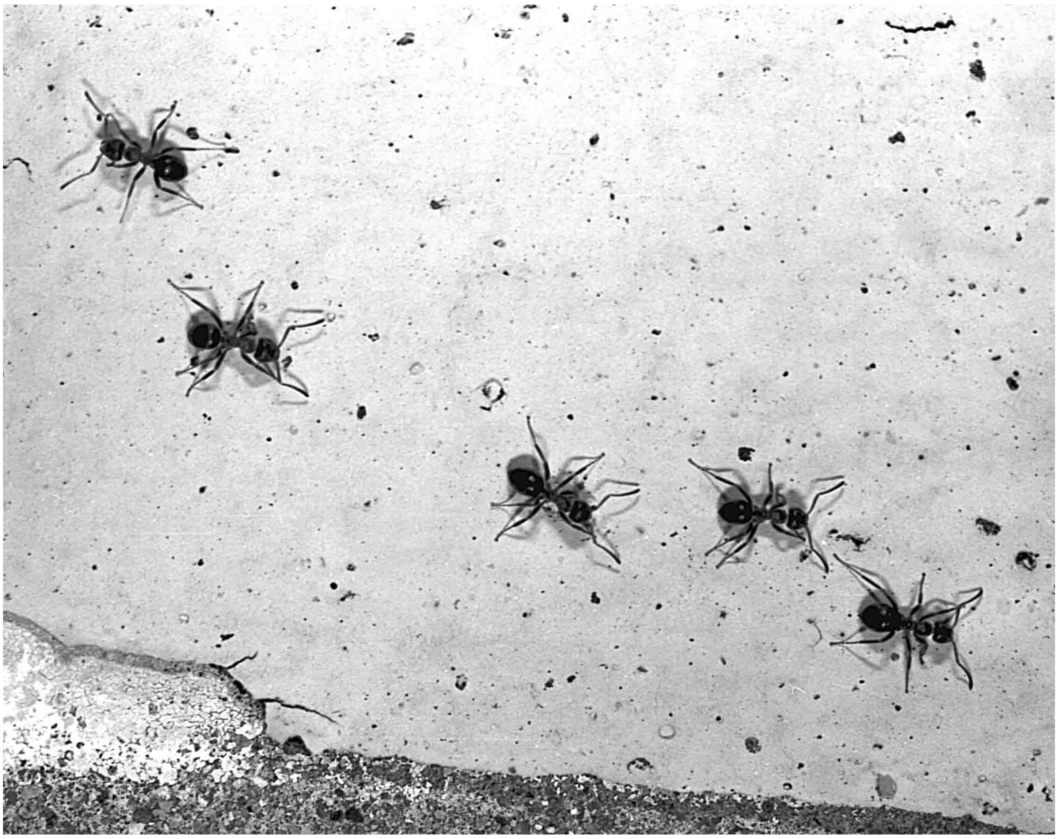
PATH RELINKING

Idea : Choose 2 solutions in a population and link them by a path in the graph of solutions + neighbourhood structure. Each intermediate node is tentatively improved by a local search.

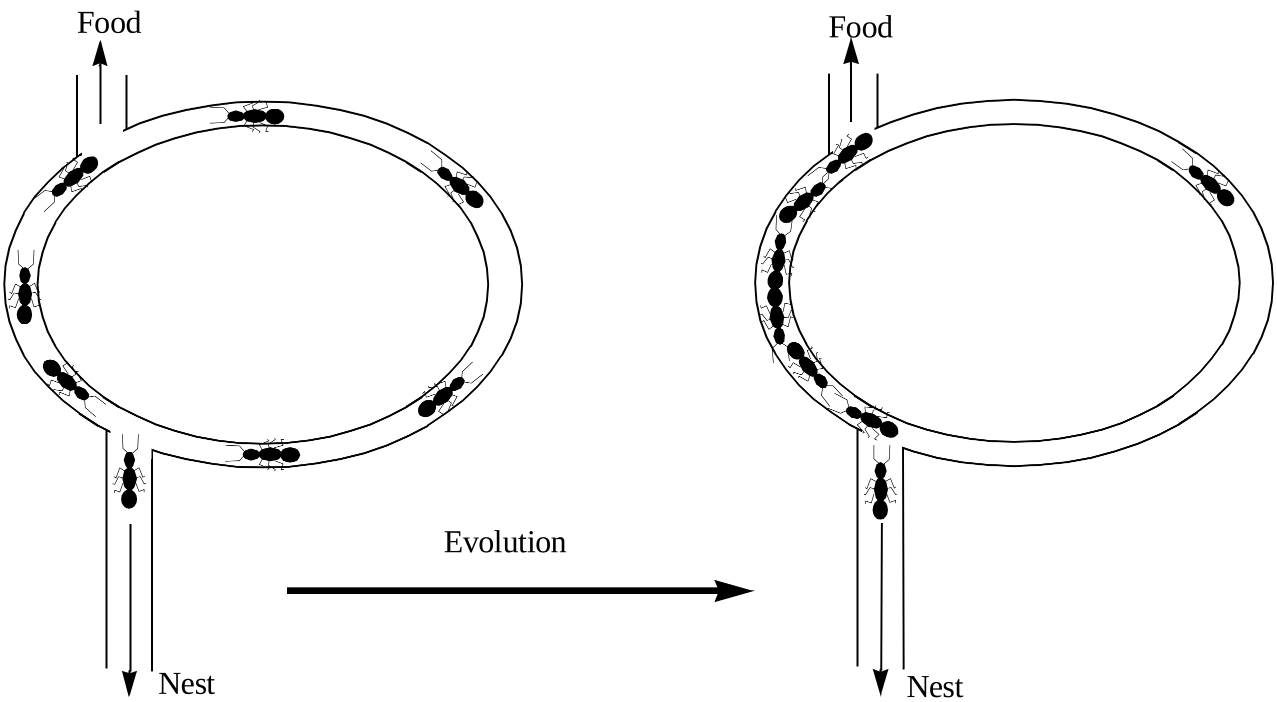
Promising applications : multi-objective optimization



ANT COLONIES

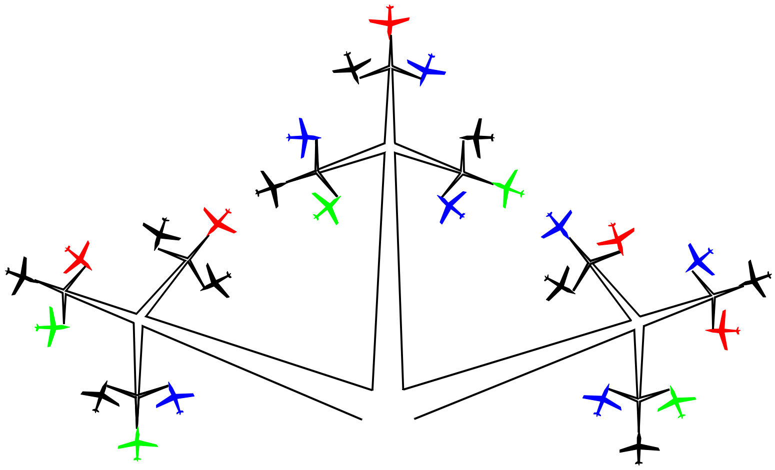


EXPERIMENT ON REAL ANTS



The nest is isolated from food source by 2 different tubes. After a while the ants use the shortest tube, because the pheromones depositted in this tube increase more rapidly.

ILLUSTRATION FOR THE GATE ASSIGNMENT PROBLEM



Which gate to assign to each aeroplane such that the total distance walk by passengers in the airport is minimized ?

→ Quadratic assignment problem.

PROBLEM MODELLING OF GATE ASSIGNMENT :

Permutation matrix $\boldsymbol{X} = (x_{ij})$: $x_{ij} = 1$ if aeroplane i is assigned to gate j, 0 otherwise

Gate

Aeroplane	1	2	3	4	5	6
1	1	0	0	0	0	0
2	0	0	0	1	0	0
3	0	1	0	0	0	0
4	0	0	0	0	0	0
5	0	0	1	0	0	0
6	0	0	0	0	1	0

ARTIFICIAL PHEROMON TRAILS

Modelling :

Matrix $T = (\tau_{ij})$

Meaning : *a-posteriori* interest to assign gate j to aeroplane i

Pheromon trail update :

$\mathbf{X}$: solution generated by an ant (permutation matrix)

$\mathbf{T} := (1 - \rho) \cdot \mathbf{T} + Q(\mathbf{X}) \cdot \mathbf{X}$, where :

$0 < \rho < 1$ is a parameter : evaporation factor

Q is a function measuring the quality of a solution $\mathbf{X}$ (e.g. $1/f(\mathbf{X})$ in case of minimization)

Pheromon usage :

An ant sequentially assign gates with probability to assign gate j to aeroplane i given by :

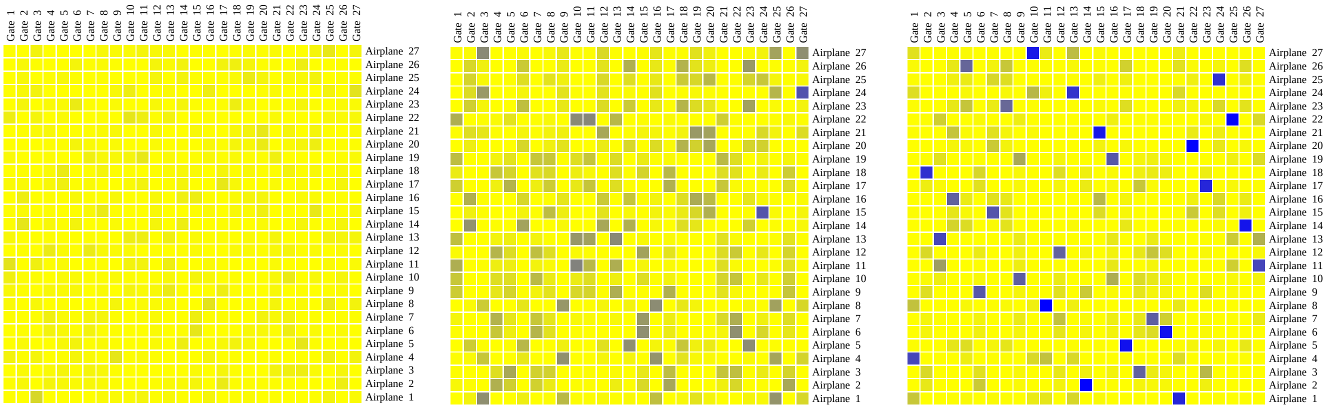
$\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}$, where :

$\alpha > 0$ and $\beta > 0$ are parameters and η_{ij} is the *a-priori* interest to assign gate j to aeroplane i

ARTIFICIAL ANTS PSEUDO-CODE (FANT FOR QAP)

```
Initialization :
   r = 1                          -- not standard
   T = 1·r

For t_max iterations :
   For each ant :
      Set X = 0                   -- problem dependent
      For each aeroplane i:
         Chose a not yet assigned gate j with probability :
            proportionnal to τ_ij -- originally: τ_ij^α · η_ij^β
         Set x_ij = 1
      End for

      Improve solution X with a local search -- Deamon actions
      If X = X* then set r = r + 1, T = 1·r
      If X better than X*, set X* = X, r = 1, T = 1·r

      Set T = T + rX + P·X* -- P : parameter. Originally: (1 – ρ)·T + Q(X)·X

   End for
End for

Return X*
```

EVOLUTION OF PHEROMON TRAILS



CONVERGENCE THEOREMS

There are convergence theorems :

Among assumptions : The minimum value of pheromon trace is not 0

Consequence : The probability of building the global optimum is not 0.

etc.

COMPLEMENT : DEALING WITH DYNAMIC PROBLEMS

Often, practical problems are dynamic (e.g. arrival of new customers, job finished, breakdown)

Possible approach : Adaptive Memory Programming (AMP)

General algorithm

Initialize memory

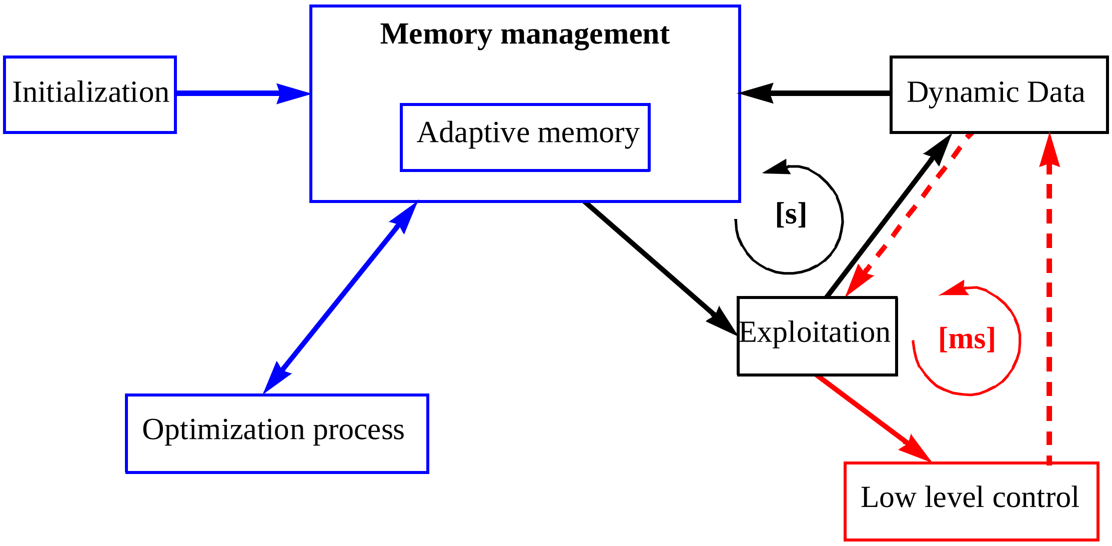
Repeat, until a stopping criterion met :

- Build a provisory solution with the help of informations in memory
- Improve provisory solution with a local search
- Update memory with informations obtained with new solution

Ant systems	Memory ≡ pheromon trails
Evolutionary algorithms, Scatter Search	Memory ≡ population of solutions
Vocabulary building	Memory ≡ pieces of solutions

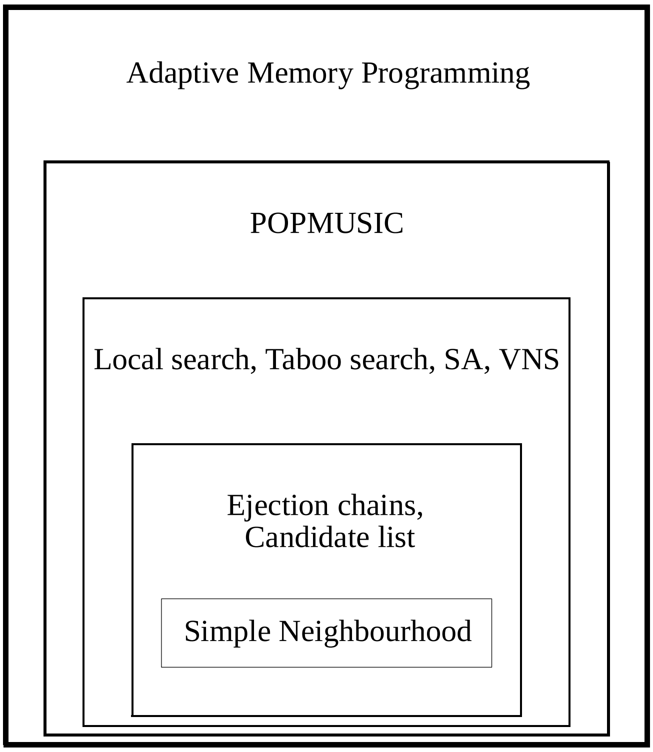
Provocative comment : Memories and building procedures for all these methods are equivalent.

AMP TECHNOLOGY

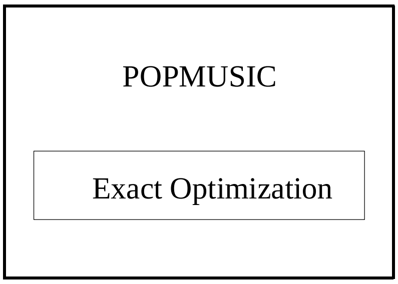


Computing power not used by real time **Real time management**

COMPLEMENT : METHODOLOGY PROPOSAL



Design of a complex method



A less complex design