Teaching Metaheuristics

Éric Taillard

eric(point)taillard(arobase)heig-vd.ch

MIC 2024, Lorient (F)

June 4. 2024

Table of Content

1. Introduction

Embarrassing questions from students

- What is the best metaheuristic?
- Which metaheuristic should I use for this problem?
- Which neighbourhood should I use for this problem?
- How many iterations are needed?
- What population size/tabu list/elite set size should I use?

Best Metaheuristic?

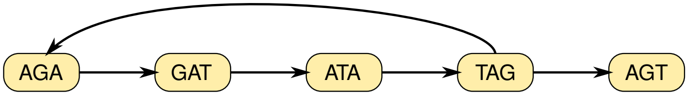
- What is a metaheuristic?
  - Simple, alternate definition:
  - Set of building blocks for designing a heuristic algorithm
  - Suggested ways of assembling these blocks
- Which is the best heuristic for this problem?
  - Answer: None
  - No Free Lunch Theorems state that no heuristic can be universally better
  - We can only design good heuristics for a given subset of problem

Which neighbourhood should I use for this problem?

Depends on problem modelling; example: **Genetic sequence to discover: AGATAGT**
Detected 3-nucleotids AGA, GAT, ATA, TAG, AGT

- de Bruijn Graphs with nodes $\equiv$ detected 3-nucleotids

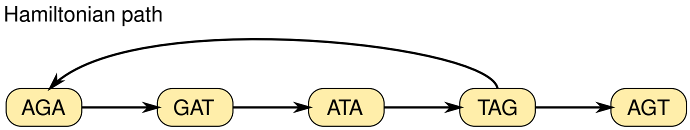
Hamiltonian path



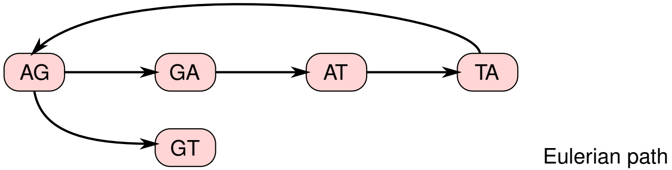
Which neighbourhood should I use for this problem?

Depends on problem modelling; example: **Genetic sequence to discover: AGATAGT**
Detected 3-nucleotids AGA, GAT, ATA, TAG, AGT

- de Bruijn Graphs with nodes ≡ detected 3-nucleotids



- de Bruijn Graphs with 3-nucleotid detected ≡ edge



Parameter Tuning

- How many iterations are needed?
  - Depends on you patience
- What population size/tabu list/elite set size should I use?
  - Use a software for automatic parameter Tuning

Parameter Tuning

- How many iterations are needed?
  - Depends on you patience
- What population size/tabu list/elite set size should I use?
  - Use a software for automatic parameter Tuning
  - Perhaps the student didn't understand how the metaheuristic works, which to him seems more like cloud sculpting, and they chose it based on its sexy name...

Parameter Tuning

- How many iterations are needed?
  - Depends on you patience
- What population size/tabu list/elite set size should I use?
  - Use a software for automatic parameter Tuning
  - Perhaps the student didn't understand how the metaheuristic works, which to him seems more like cloud sculpting, and they chose it based on its sexy name...
    - "I want to implement a Wild Wombat Tango (WWT) procedure"

Parameter Tuning

- How many iterations are needed?
  - Depends on you patience
- What population size/tabu list/elite set size should I use?
  - Use a software for automatic parameter Tuning
  - Perhaps the student didn’t understand how the metaheuristic works, which to him seems more like cloud sculpting, and they chose it based on its sexy name...
    - "I want to implement a Wild Wombat Tango (WWT) procedure"
  - Important to demonstrate how to design an effective heuristic from scratch in a simple manner

Parameter Tuning

- How many iterations are needed?
  - Depends on you patience
- What population size/tabu list/elite set size should I use?
  - Use a software for automatic parameter Tuning
  - Perhaps the student didn't understand how the metaheuristic works, which to him seems more like cloud sculpting, and they chose it based on its sexy name...
    - "I want to implement a Wild Wombat Tango (WWT) procedure"
  - Important to demonstrate how to design an effective heuristic from scratch in a simple manner
  - Provide basic procedural codes

Reference Books for this Presentation

- É. D. Taillard Design of Heuristic Algorithms for Hard Optimization with Python Codes for the Travelling Salesman Problem Springer, 2023
- É. D. Taillard Design of Heuristic Algorithms for Hard Optimization with C Codes for the Travelling Salesman Problem
- Beamer Latex source files, including all figures, tables, algorithms, ILP model of the book
- Open Access CC-BY
- To lighten the slides, the references are grouped at the end

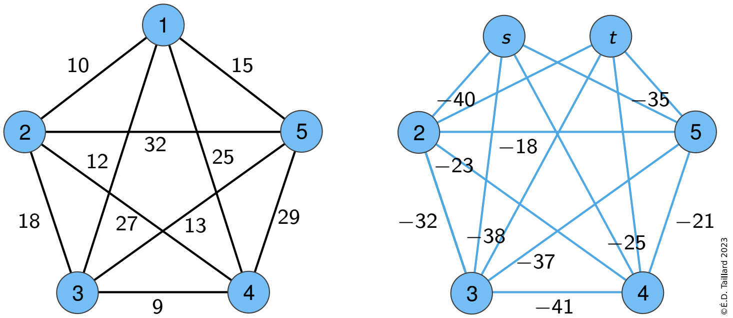
Alternate Definition of Metaheuristics

Set of building blocks for designing a heuristic algorithm

- Problem Modelling (not specific to metaheuristics!)
  - Classification, simulation
  - Mono vs multi-objective optimization
  - Problem decomposition
- Solution Building
- Solution Improvement
  - Sub-problem optimization
  - Matheuristics
  - POPMUSIC
- Learning
  - Construction Learning: Artificial Ant Colony
  - Improvement Learning: Tabu Search
  - Learning with Solutions: Genetic Algorithms, Scatter Search, Particle Swarm

Iconic Problem: the TSP

Travelling Salesman Problem (TSP) $\propto$ Elementary Shortest Path



- Data: n cities, distance matrix $D = (d_{ij})$
- Solution: Permutation π of the n cities
- Objective: $\min_\pi \sum_{i=1}^{n-1} d_{\pi_i \pi_{i+1}} + d_{\pi_n \pi_1}$

2. Constructive methods

Kruskal Algorithm for Minimum Spanning Tree

Input : A network with set E of edges
Weight $w(e) \quad \forall e \in E$
Result : Minimum Spanning Tree T

```
Start with an empty tree T
R ← E                                         // Edges that can be potentially added to T
while R ≠ ∅ do
    Choose e′ ∈ R minimizing w(e′)
    T ← T ∪ e′                                // Include e′ in the partial tree T
    Remove from R the edges that cannot be added any more to T (degree 3, cycle)
```

Dijkstra's algorithm for computing shortest paths is very similar

Greedy Constructive Method

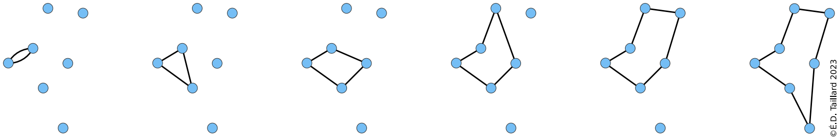
Input : Set E of elements constituting a solution
Incremental cost function $c(s, e)$
Result : Complete solution s

Start with a trivial partial solution s (generally $\varnothing$)
$R \leftarrow E$ // Elements that can be potentially added to s
while $R \neq \varnothing$ **do**
  Choose $e' \in R$ optimizing $c(s, e')$
  $s \leftarrow s \cup e'$ // Include e' in the partial solution s
  Remove from R the elements that cannot be added any more to s

Apply the same approach to a difficult problem as the one that works for a simple problem

Least Cost Insertion for the TSP

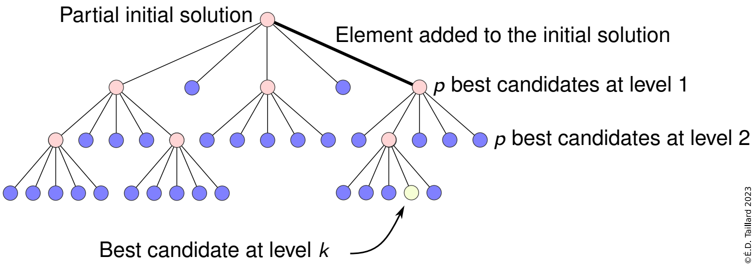
- Start from a partial tour containing a single city
- Element e à to add: a city
- Incremental cost: Minimum detour to add e to the partial tour
- Choose the city with the lowest incremental cost



Seems to work not too bad for the TSP

Beam Search

- Imitate implicit enumeration
- Avoid a myopic greedy choice by examining k forward insertions
- Avoid exponential explosion by keeping only the p best candidate at each level
- $c(s, e)$: Cost of best candidate in branch e at the last level



Beam search plays an important role in AI

3. Local Search

Bellman-Ford Algorithm for Shortest Path

Data : Directed network $R = (V, E, w)$ given with an arc list, a starting node s
Result : Immediate predecessor $pred_j$ of j on a shortest path from s to j with its length λ_j, $\forall j \in V$, or: warning message of the existence of a negative length circuit

forall $i \in V$ **do**
    $\lambda_i \leftarrow w(s, i) \quad (\infty$ if arc $(i, j) \notin E)$
    $pred_i \leftarrow s$

$k \leftarrow 0$ // Step counter

$continue \leftarrow true$ // At least one λ modified at last step

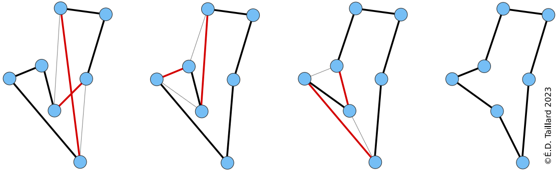
while $k < |V|$ *and continue* **do**
    $continue \leftarrow false$
    $k \leftarrow k + 1$
    forall *arc* $(i, j) \in E$ // Check if a better path can be identified
    do
        if $\lambda_j > \lambda_i + w(i, j)$ // Improvement found: modify the solution
        then
            $\lambda_j \leftarrow \lambda_i + w(i, j)$
            $pred_j \leftarrow i$
            $continue \leftarrow true$

if $k = |V|$ **then**
    Warning: there is a negative length circuit that can be reached from s

Local search

Bellman-Ford works fine for finding shortest paths

- It's a local improvement technique, like the Simplex algorithm
- Start with a solution obtained with a simple method
- Improve it with local modifications



Two edges are replaced by two others whose sum of lengths is smaller
Imitate a gradient-like method for a non-differentiable function

Local Search Frame: Best Improvement

Input : Solution s, neighbourhood specification $N(\cdot)$, fitness function $f(\cdot)$ to minimize.

Result : Improved solution s

repeat
    $end \leftarrow$ true
    $best\_neighbour\_value \leftarrow \infty$
    forall $s' \in N(s)$ **do**
        if $f(s') < best\_neighbour\_value$ **then** A better neighbour is found
            $best\_neighbour\_value \leftarrow f(s')$
            $best\_neighbour \leftarrow s'$
    if $best\_neighbour\_value < f(s)$ **then** Move to the improved solution
        $s \leftarrow best\_neighbour$
        $end \leftarrow$ false
until end

Pareto Local Search for Multi-Objective Optimization

Neighbourhood_evaluation

Input : Solution s; neighbourhood $N(\cdot)$ objective functions $\vec{f}(\cdot)$

Result : Approximation of Pareto set P completed with neighbours of s

```
forall s′ ∈ N(s) do
    Update_Pareto(s′, f⃗(s′))
```

Update_Pareto

Input : Solution s, objective values $\vec{v}$

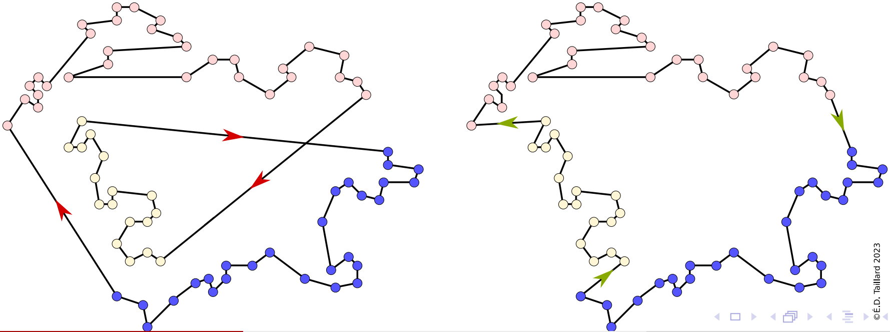
Result : Updated Pareto set P

```
if (s, v⃗) either dominates a solution of P or P = ∅ then
    From P, remove all the solutions dominated by (s, v⃗)
    P ← P ∪ (s, v⃗)
    Neighbourhood_evaluation(s)
```

TSP 3-opt move

Replace 3 edges by 3 others

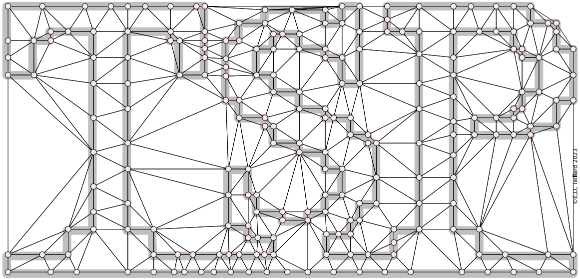
- $(i \to s_i), (j \to s_j), (k \to s_k)$ replaced by: $(i \to s_j), (j \to s_k), (k \to s_i)$
- Respect the edge orientation on the other edges (not the case for 2-opt)
- Cons: Algorithmic complexity



Limitation of Neighbourhood Size: Candidate List, Granular Search

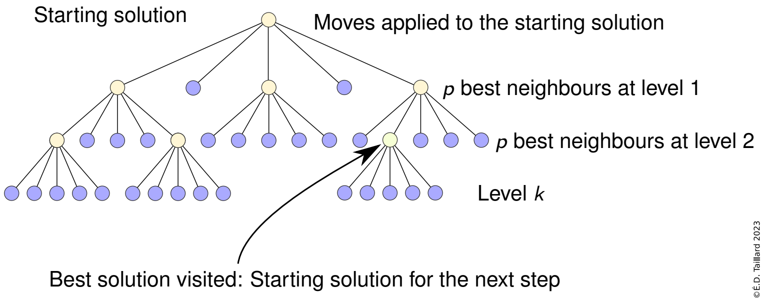
Idea: select a subset of potentially interesting neighbour solutions

- Example for the Euclidean TSP
  - Keep only the edges of the Delaunay triangulation
- Generate the edges with fast POPMUSIC



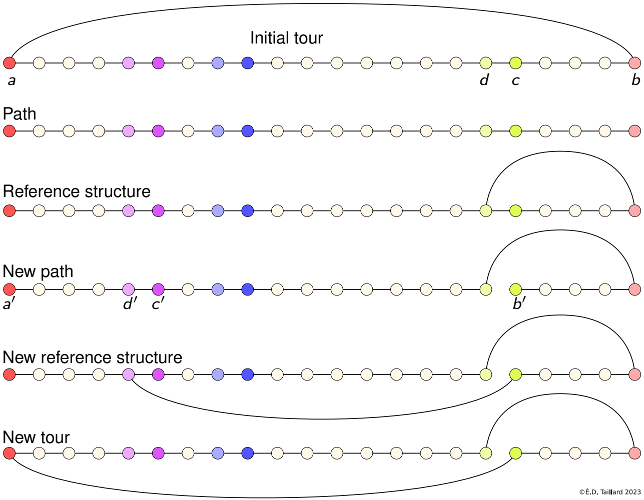
Neighbourhood extension: Filter and Fan

Imitate Beam Search, but working with a Neighbourhood

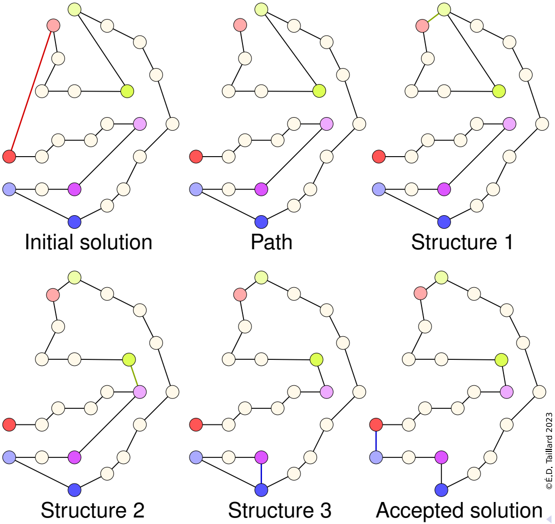


Note: It might be interesting to extend a previously limited neighbourhood

Ejection Chain for the TSP: Lin-Kernighan Neighbourhood

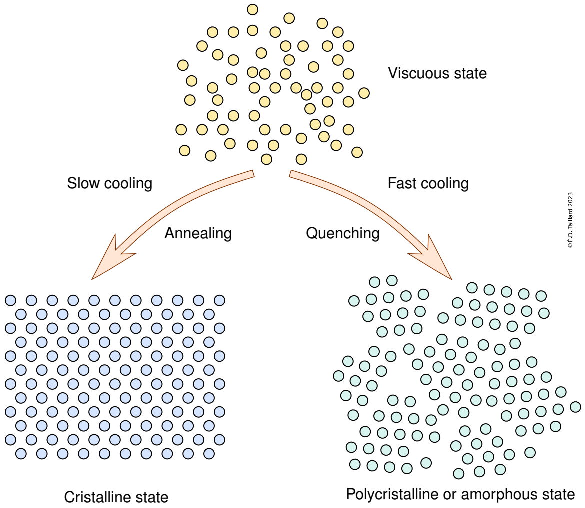


Ejection Chain: Lin-Kernighan Neighbourhood



4. Randomized Methods

Annealing and Quenching Physical Process



Simulated Annealing: Randomized Local Search

Input : Initial solution s; fitness function f to minimize; neighbourhood structure M, parameters T_{init}, $T_{end} < T_{init}$ and $0 < \alpha < 1$

Result : Modified solution s

$T \leftarrow T_{init}$
while $T > T_{end}$ **do**
  Randomly generate $m \in M$
  $\Delta = f(s \oplus m) - f(s)$
  Randomly generate $0 < u < 1$
  if $\Delta < 0$ *or* $e^{-\Delta/T} > u$ **then** m is accepted
    $s \leftarrow s \oplus m$
  $T \leftarrow \alpha \cdot T$

Metaheuristics Similar to SA

- Threshold Accepting
- Great Deluge
- Demon Algorithm
- Generalization: Noising Methods

Strategies Combining Simple Blocks

- Variable Neighbourhood Search
  - Basic strategic oscillation
  - Perturb the solution by applying a move picked at random in various neighbourhoods
  - Apply a local improvement method (fixed neighbourhood)
- Greedy Randomized Adaptive Search Procedure
  - Build a solution with random choices
  - Apply a local improvement method

Greedy Randomized Adaptive Search Procedure (GRASP)

Input : Set E of elements constituting a solution; incremental cost function $c(s,e)$; fitness function f to minimize, parameters I_{max} and $0 \leqslant \alpha \leqslant 1$, improvement method *local_search*

Result : Complete solution s^*

```
f* ← ∞
for I_max iterations do
    Initialize s to a trivial partial solution
    R ← E                                         // Elements that can be added to s
    while R ≠ ∅ do
        Find c_min = min_{e∈R} c(s, e) and c_max = max_{e∈R} c(s, e)
        Choose randomly, uniformly e' ∈ R such that c_min ⩽ c(s, e') ⩽ c_min + α(c_max − c_min)
        s ← s ∪ e'                                // Include e' in the partial solution s
        Remove from R the elements that cannot be added any more to s
    s' ← local_search(s)                          // Find the local optimum associated with s
    if f* > f(s') then
        f* ← f(s')
        s* ← s'
```

5. Metaheuristic Learning Techniques

Construction Learning: Artificial Ants

Simplified Idea: GRASP with Learning

- Compute a statistics τ_e for each element e constituting a potential solution (artificial pheromone)
  - Running average depending on the number of times e appears in previously generated solutions, fitness, ...
  - MAX-MIN Ant System: maintain $\tau_{min} \leqslant \tau_e \leqslant \tau_{max}$
- Instead of choosing any element e such that $c_{min} \leqslant c(s,e) \leqslant c_{min} + \alpha(c_{max} - c_{min})$, choose e with a probability depending on τ_e and $c(s,e)$
- Add parameters (α, β) for balancing a priori interest $c(s,e)$ and a posteriori interest τ_e
- Forgetting is very important in machine learning, in order to generalize and avoid overfitting; add parameter ρ
- Other options: m solutions built in parallel, choice of solutions used to update τ

MAX-MIN Ant System

Input : Set E of elements constituting a solution; incremental cost function $c(s, e) > 0$; fitness function f to minimize, parameters $I_{max}, m, \alpha, \beta, \tau_{min}, \tau_{max}, \rho$ and improvement method $a(\cdot)$

Result : Solution s^*

$f^* \leftarrow \infty$

for $\forall e \in E$ **do**

  $\tau_e \leftarrow \tau_{max}$

for I_{max} *iterations* **do**

  for $k = 1 \dots m$ **do**

    Initialize s as a trivial, partial solution

    $R \leftarrow E$ // Elements that can be added to s

    while $R \neq \varnothing$ **do** Build a new solution

      Randomly choose $e \in R$ with a probability proportional to $\tau_e^\alpha \cdot c(s, e)^\beta$ // Ant colony formula

      $s \leftarrow s \cup e$

      From R, remove the elements that cannot be added any more to s

    $s_k \leftarrow a(s)$ // Find the local optimum s_k associated with s

    if $f^* > f(s_k)$ **then** Update the best solution found

      $f^* \leftarrow f(s_k)$

      $s^* \leftarrow s_k$

  for $\forall e \in E$ **do** Pheromone trail evaporation

    $\tau_e \leftarrow (1 - \rho) \cdot \tau_e$

  $s_b \leftarrow$ best solution from $\{s_1, \dots, s_m\}$

  for $\forall e \in s_b$ **do** Update trail, maintaining it between the bounds

    $\tau_e \leftarrow max(\tau_{min}, min(\tau_{max}, \tau_e + 1/f(s_b)))$

Fast Ant System (FANT)

Simplified artificial ant system:

- Only 2 explicit parameters
- An implicit parameter is auto-adaptative
- No a priori interest

Local Search Learning: Tabu Search

- Local search with best move policy
- Allow degrading moves
- Use a memory to avoid visiting cyclically a subset of solutions
  - Forbid to come back to a solution already visited (the solution is *tabu*)
  - Forbid to perform the reverse of a move recently used
  - Penalize frequently performed moves
  - Force the use of moves never performed for a long time
- Oblivion
  - Remove a prohibition after a certain number of iterations
- A lot of other strategies suggested in the original work (aspiration, candidate list, oscillations, . . .)

Tabu Search

Input : Solution s, set M of moves, fitness function $f(\cdot)$ to minimize, parameters I_{max}, d

Result : Improved solution s^*

$s^* \leftarrow s$

for I_{max} *iterations* **do**

    $best\_neighbour\_value \leftarrow \infty$

    forall $m \in M$ *(such that* m *(or* $s \oplus m$*) is not marked as taboo)* **do**

        if $f(s \oplus m) < best\_neighbour\_value$ **then**

            $best\_neighbour\_value \leftarrow f(s \oplus m)$

            $m^* \leftarrow m$

    if $best\_neighbour\_value < \infty$ **then**

        Mark $(m^*)^{-1}$ (or s) as taboo for the next d iterations

        $s \leftarrow s \oplus m^*$

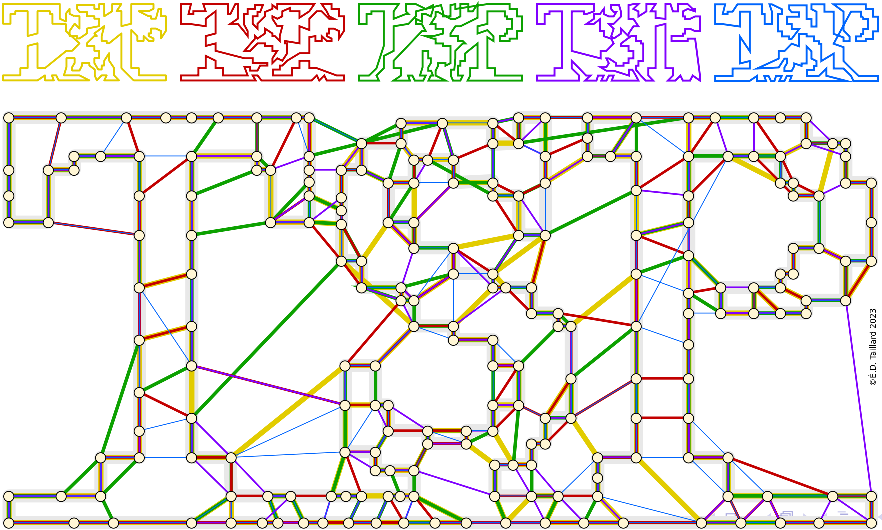
        if $f(s) < f(s^*)$ **then**

            $s^* \leftarrow s$

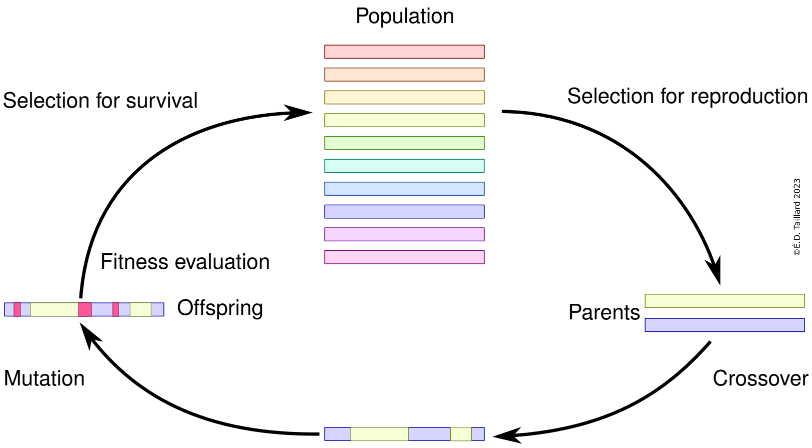
    else

        Error message: d too large: no move allowed!

Population Learning



Generational Loop in an Evolutionary Algorithm



Population Management Principles

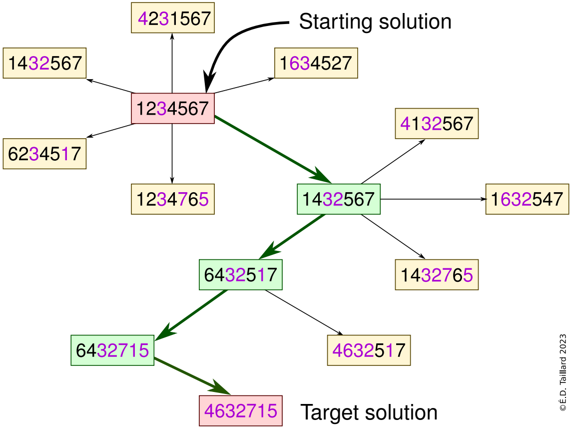
- Keep a subset of elite solutions in the population
- Introduce a diversity measure between solutions and keep solutions as scattered as possible in the population
- Exploitation of the population
  - Mix 2 solutions: genetic crossover
  - Mix several solutions: scatter search
  - Apply a local search to the new created solutions
  - Go from a starting solution to a target solution using a neighbourhood: path relinking

Getting an Offspring

Depends on the problem, but technically possible!

Getting an Offspring: Scatter Search Extension

Exploiting a Population of Solutions: Path Relinking



GRASP with Path Relinking

Input : GRASP procedure (with local search LS and parameter $0 \leqslant \alpha \leqslant 1$), parameters I_{max} and μ

Result : Population P of solutions

$P \leftarrow \varnothing$

while $|P| < \mu$ **do**

  $s \leftarrow GRASP(\alpha, \mathrm{LS})$

  if $s \notin P$ **then**

    $P \leftarrow P \cup s$

for I_{max} *iterations* **do**

  $s \leftarrow GRASP(\alpha, \mathrm{LS})$

  Randomly draw $s' \in P$ Apply a path relinking method between s and s'; identifying the best solution s'' of the path

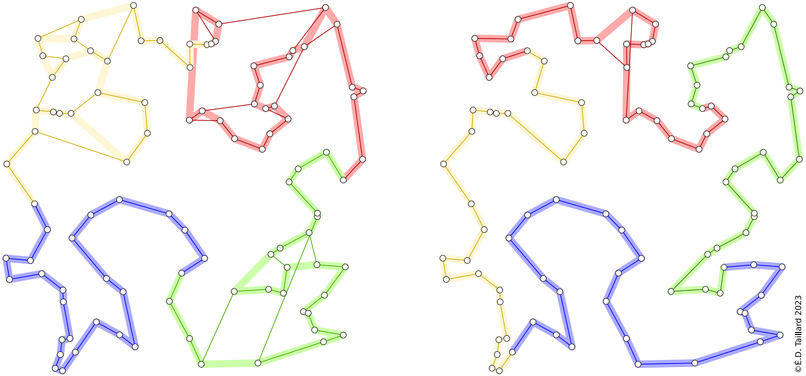
  if $s'' \notin P$ *and* s'' *is strictly better than a solution of* P **then**

    s'' replaces the most different solution of P which is worse than s''

6. Decomposition Methods

Improvement of a TSP Tour

Decompose the tour into sub-paths containing approximately r cities
Optimize each sub-path with a good quality method
Restart by considering overlapping sub-paths



POPMUSIC Frame

Input : Initial solution s composed of q disjoint parts $s_1, \ldots, s_q$; sub-problem improvement method

Result : Improved solution s

$U = \{s_1, \ldots, s_q\}$

while $U \neq \varnothing$ **do**

  Select $s_g \in U$ // s_g: Seed part

  Build a sub-problem R composed of the r parts of s the closest to s_g

  Tentatively optimize R

  if R *is improved* **then**

    Update s

    From U, remove the part no longer belonging to s

    In U, insert the parts composing R

  else R not improved

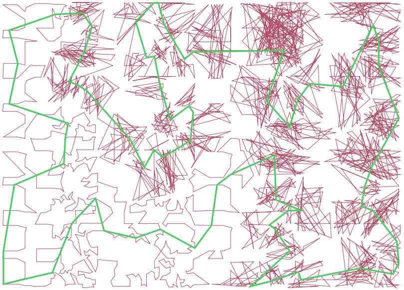
    Remove s_g from U

Other Frames Related to POPMUSIC

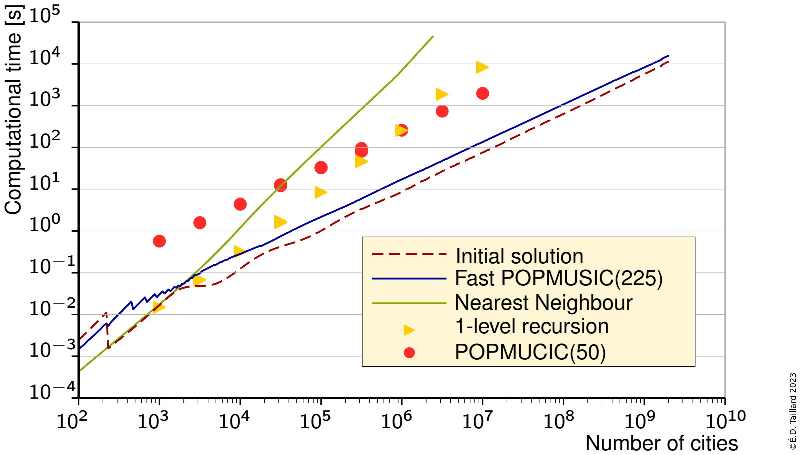
- Destroy and Repair
  - Large Neighbourhood Search (LNS)
  - Adaptive LNS: Neighbouhood Selection
  - Variable Neighbourhood Search
- Magnifying Glass Method
- Matheuristics
- . . .

POPMUSIC for the TSP: Empirical Complexity in $O(n^{1.57})$

- Select a sample of $O(n^{0.56})$ cities
- Find a good tour on the sample with Lin-Kernighan neighbourhood
- Group all the cities into a number of clusters equals to the sample
- Optimize the tour with 2-opt neighbourhood by considering 2 successive clusters at a time
- Re-optimize the tour with POPMUSIC (all subsets of 50 successive cities are LK optimum)

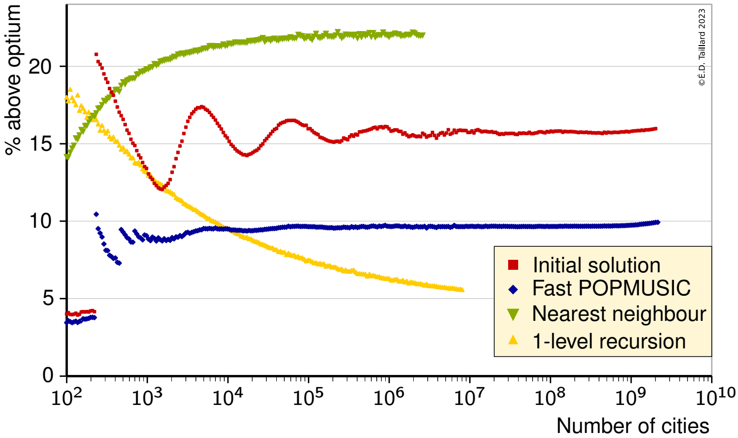


POPMUSIC Empirical Complexity



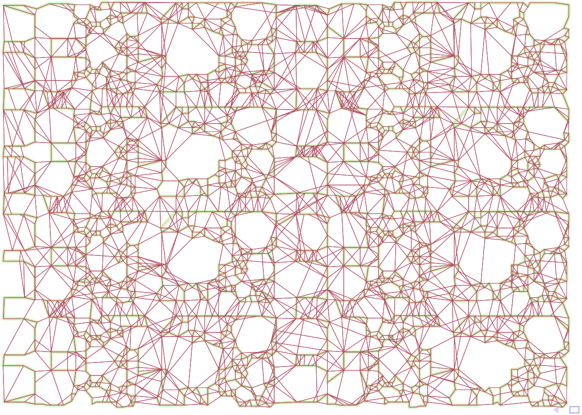
Solution Quality for the TSP with toroidal distances

Fast POPMUSIC: Initial solution obtained recursively, $2 \times n/225$ sub-problem optimization
The Lin-Kernighan method used to optimize sub-path produces solutions 4% above optimum



Union 20 POPMUSIC Solutions

Optimum solution in green
Method now included in LKH solver for filtering the potential edges retained



Conclusions

Missing Chapters

- Merging machine learning and metaheuristics
  - Metaheuristic parameter tuning, hyper-heuristics
  - Direct optimization: Large training times
  - Limited instance size
  - After training: relatively good solutions obtained rapidly
- Quantum computing
  - Good solutions of very specific sparse QUBO instances obtained faster than classical heuristics running on classical machine

Questions ?

- Alvim A.C.F., Taillard É.D.: POPMUSIC for the World Location Routing Problem. EURO Journal on Transportation and Logistics **2**(3), 231–254 (2013). https://doi.org/10.1007/s13676-013-0024-2
- Applegate D.L., Bixby R.E., Chvátal V., Cook W.J.: Concorde: A code for solving Traveling Salesman Problems. https://github.com/matthelb/concorde (1999). Accessed 16 June 2022
- Battiti R., Tecchiolli G.: The Reactive Tabu Search, ORSA Journal on Computing **6**, 126–140 (1994). https://doi.org/10.1287/ijoc.6.2.126
- Cavaliere F., Accorsi L., Laganà D., Musmanno R., Vigo D.: An efficient heuristic for very large-scale vehicle routing problems with simultaneous pickup and delivery. Transportation Research Part E: Logistics and Transportation Review **186**, (2024). https://www.sciencedirect.com/science/article/pii/S1366554524001418
- Černý V.: Thermodinamical Approach to the Traveling Salesman Problem: An efficient Simulation Algorithm. Journal of Optimization Theory and Applications **45**(1), 41–51 (1985). https://doi.org/10.1007/BF00940812
- Charon I., Hudry O.: The Noising Method: a New Method for Combinatorial Optimization. Operations Research Letters **14**(3), 133–137 (1993). https://doi.org/10.1016/0167-6377(93)90023-A
- Colorni A., Dorigo M., Maniezzo V.: Distributed Optimization by Ant Colonies. In: Actes de la première conférence européenne sur la vie artificielle, pp. 134–142, Paris. Elsevier, (1991)

- Cook S.A.: The Complexity of Theorem-Proving Procedures. In: Proceedings of the third annual ACM symposium on Theory of computing 151–158. ACM (1971). https://doi.org/10.1145/800157.805047
- Croes G.A.: A Method for Solving Traveling Salesman Problems. Operations Research **6**, 791–812 (1958). https://doi.org/10.1287/opre.6.6.791
- Deneubourg J., Goss S., Pasteels J., Fresneau D., Lachaud J.: Self-Organization Mechanisms in Ant Societies (II) : Learning in Foraging and Division of Labor. In: Pasteels J., et al. (eds.) From Individual to Collective Behavior in Social Insects, Experientia supplementum **54**, 177–196. Birkhäuser, Basel (1987)
- Dorigo M., Gambardella L.M.: Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation **1**(1), 53–66 (1997). https://doi.org/10.1109/4235.585892
- Dueck G.: New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. Journal of Computational Physics **104**(1), 86–92 (1993). https://doi.org/10.1006/jcph.1993.1010
- Dueck G., Scheuer T.: Threshold accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing. Journal of Computational Physics **90**(1), 161–175(1990). https://doi.org/10.1016/0021-9991(90)90201-B

- Duin C., Voß S.: The Pilot Method: A Strategy for Heuristic Repetition with Application to the Steiner Problem in Graphs. Networks **34**, 181-191 (1999). https://doi.org/10.1002/(SICI)1097-0037(199910)34:3%3C181::AID-NET2%3E3.0.CO;2-Y
- Feo T.A., Resende M.G.C.: Greedy Randomized Adaptive Search Procedure. Journal of Global Optimization **6**, 109–133 (1995). https://doi.org/10.1007/BF01096763
- Furcy D., Koenig S.: Limited Discrepancy Beam Search. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 125–131 (2005)
- Garey M.R., Johnson D.S.: Computers and Intractability — A Guide to the Theory of NP-Completeness. W. H. Freeman & Co. (1979)
- Gilli M., Këllezi E., Hysi H.: A Data-Driven Optimization Heuristic for Downside Risk Minimization. Journal of Risk **8**(3), 1–19 (2006)
- Glover F.: Heuristics for Integer Programming Using Surrogate Constraints. Decision Sciences **8**(1), 156–166 (1977). https://doi.org/10.1111/j.1540-5915.1977.tb01074.x
- Glover F.: Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research **13**(5), 533–549 (1986). https://doi.org/10.1016/0305-0548(86)90048-1
- Glover F.: Tabu Search — part I. ORSA Journal on Computing **1**(3), 190–206 (1989). https://doi.org/10.1287/ijoc.1.3.190

- Glover F.: Tabu Search — part II. ORSA Journal on Computing **2**(1), 4–32 (1990). https://doi.org/10.1287/ijoc.2.1.4
- Glover F.: Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems. Discrete Applied Mathematics **65**, 223–253 (1996). https://doi.org/10.1016/0166-218X(94)00037-E
- Glover F.: Tabu Search and Adaptive Memory Programming — Advances, Applications and Challenges. In: Barr R.S., Helgason R.V., Kennington J.L. (eds.) Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies. pp. 1–75. Springer, Boston (1997). https://doi.org/10.1007/978-1-4615-4102-8_1
- Glover F.: A Template for Scatter Search and Path Relinking. In: Hao J.K., Lutton E., Ronald E., Schoenauer M., Snyers D. (eds.) Artificial Evolution, Lecture Notes in Computer Science **1363**, 13–54 (1998). https://doi.org/10.1007/BFb0026589
- Glover F., Laguna M.: Tabu Search. Kluwer, Dordrecht (1997)
- Greistorfer P., Staněk R., Maniezzo V.: The Magnifying Glass Heuristic for the Generalized Quadratic Assignment Problem. Metaheuristic International Conference (MIC'19) proceedings, Cartagena, Columbia (2019)

- Helsgaun K.: An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. European Journal of Operational Research **126**(1), 106–130 (2000). https://doi.org/10.1016/S0377-2217(99)00284-2
- Helsgaun K.: Using POPMUSIC for Candidate Set Generation in the Lin-Kernighan-Helsgaun TSP Solver. Department of Computer Science, Roskilde University, Denmark (2018)
- Holland J.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Harbor (1975)
- Jarník V.: O jistém problému minimálním. (Z dopisu panu O. Borůvkovi [On a certain problem of minimization (from a letter to O. Borůvka)], in Czech. Práce moravské přírodovědecké společnosti **6**(4), 57–63 (1930). http://dml.cz/dmlcz/500726
- Jarník V.: O minimálních grafech, obsahujících n daných bodů [On minimal graphs containing n given points], in Czech. Časopis pro Pěstování Matematiky a Fysiky **63**(8) 223–235 (1934). https://doi.org/10.21136/CPMF.1934.122548
- Jovanovic R., Tuba M., Voß S.: Fixed Set Search Applied to the Traveling Salesman Problem. In: Blum C., Gambini Santos H., Pinacho-Davidson P., Godoy del Campo J. (eds.) Hybrid Metaheuristics. Lecture Notes in Computer Science, vol. 11299, pp. 63–77 Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05983-5_5

- Kaufman L., Rousseeuw P.J.: Clustering by means of Medoids. In: DodgeY. (ed.) Statistical Data Analysis Based on the L_1-Norm and Related Methods, pp. 405–416 North-Holland, Amsterdam (1987)
- Kennedy J., Eberhart R.C.: Particle Swarm Optimization. IEEE International Conference on Neural Networks. vol. IV, pp. 1942–1948 Piscataway, NJ. IEEE Service Center, Perth (1995). https://doi.org/10.1109/ICNN.1995.488968
- Kirkpatrick S., Gelatt C.D., Vecchi M.P.: Optimization by Simulated Annealing. Science **220** (4598), 671–680 (1983). https://doi.org/10.1126/science.220.4598.671
- Laguna M., Martí R.: GRASP and Path Relinking for 2-layer Straight Line Crossing Minimization. INFORMS Journal on Computing **11**(1), 44–52 (1999). https://doi.org/10.1287/ijoc.11.1.44
- Lin S., Kernighan B. W.: An Effective Heuristic Algorithm for the Traveling-Salesman Problem. Operations Research **21**(2), 498–516 (1973). https://www.jstor.org/stable/169020
- Lones M.: Mitigating Metaphors: A Comprehensible Guide to Recent Nature-Inspired Algorithms. SN Computer Science **1**(49), (2020). https://doi.org/10.1007/s42979-019-0050-8
- Lowerre B.: The Harpy Speech Recognition System. Ph. D. Thesis, Carnegie Mellon University (1976)

- López-Ibáñez M., Dubois-Lacoste J., Pérez Cáceres L., Birattari M., Stützle T.: The Irace Package: Iterated Racing for Automatic Algorithm Configuration. Operations Research Perspectives **3**, 43–58 (2016). https://doi.org/10.1016/j.orp.2016.09.002
- Martello S., Toth P.: Knapsack Problems — Algorithms and Computer Implementations. Wiley, Chichester (1990)
- Hansen P., Mladenović N.: An Introduction to Variable Neighborhood Search. In: Voß S. , Martello S., Osman I.H., Roucairol C. (eds.) Meta-heuristics : Advances and Trends in Local Search Paradigms for Optimization, pp. 422–458, Kluwer, Dordrecht (1999). `https://doi.org/10.1007/978-1-4615-5775-3_30`
- Moscato P: Memetic Algorithms: A Short Introduction. In: CorneD., Glover F., Dorigo M. (eds.) New Ideas in Optimisation, pp. 219–235, McGraw-Hill, London (1999)
- Or I.: Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking. Ph. D. Thesis, Northwestern University (1976)
- Osterman C., Rego C.: A k-level Data Structure for Large-scale Traveling Salesman Problems. Annals of Operations Research **244**(2), 1–19 (2016). `https://doi.org/10.1007/s10479-016-2159-7`
- Paquete L., Chiarandini M., Stützle T.: Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study. In: Gandibleux, X., Sevaux, M., Sörensen, K., T'kindt, V. (eds.)

Multiobjective Optimisation. Lecture Notes in Economics and Mathematical Systems, vol. 535, pp. 177–200 Springer (2004). https://doi.org/10.1007/978-3-642-17144-4_7

Rechenberg I.: Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart (1973). https://doi.org/10.1002/fedr.19750860506

Reinelt G.: TSPLIB — A T.S.P. Library. (1990). http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95. Accessed 16 June 2022

Resende M.G.C., Riberio C.C.: Optimization by GRASP: Greedy Randomized Adaptive Search Procedures. Springer, New-York, (2016). https://doi.org/10.1007/978-1-4939-6530-4

Shaw P.: Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. 4. International Conference of Principles and Practice of Constraint Programming, pp. 417–431. Springer-Verlag (1998). https://doi.org/10.1007/3-540-49481-2_30

Sniedovich M., Voß S.: The Corridor Method: a Dynamic Programming Inspired Metaheuristic. Control and Cybernetics **35**(3), 551–578 (2006). http://eudml.org/doc/209435

Sörensen K., Seveaux M.: MA|PM: Memetic Algorithms with Population Management. Computers & Operations Research **33**, 1214–1225 (2006). https://doi.org/10.1016/j.cor.2004.09.011

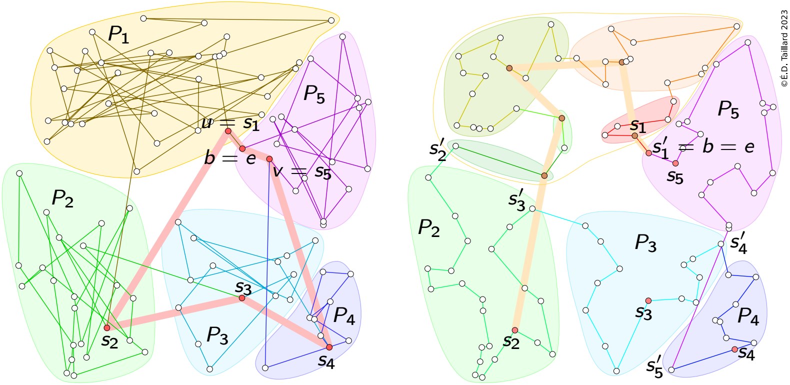
Stützle T., Hoos H.H.: MAX MIN Ant System. Future Generation Computer Systems **16**(8), 889–914 (2000). https://doi.org/10.1016/S0167-739X(00)00043-1

- Taillard É.D.: Parallel Iterative Search Methods for Vehicle Routing Problems. Networks **23**(8), 661–673 (1993). https://doi.org/10.1002/net.3230230804
- Taillard É.D.: La programmation à mémoire adaptative et les algorithmes pseudo-gloutons: nouvelles perspectives pour les méta-heuristiques. HDR Thesis, Université de Versailles-Saint-Quentin-en-Yvelines (1998)
- Taillard É.D.: Heuristic Methods for Large Centroid Clustering Problems. J. Heuristics **9**(1), 51–73 (2003). https://doi.org/10.1023/A:1021841728075
- Taillard É.D.: Tutorial : Few guidelines for analyzing methods. Metaheuristic International Conference (MIC'05) proceedings, Wien, Austria (2005)
- Taillard É.D.: A Linearithmic Heuristic for the Travelling Salesman Problem. European Journal of Operational Research **297**(2), 442–450 (2022). https://doi.org/10.1016/j.ejor.2021.05.034
- Taillard É.D.: Design of Heuristic Algorithms for Hard Optimization — with Python Codes for the Travelling Salesman Problem. Springer, Cham, Switzerland (2023). http://dx.doi.org/10.1007/978-3-031-13714-3
- Taillard É.D., Waelti P., Zuber J.: Few Statistical Tests for Proportions Comparison. European Journal of Operational Research **185**(3), 1336–1350 (2008). https://doi.org/10.1016/j.ejor.2006.03.070

- Tassef B., Albash T., Morrel Z., Vuffray M., Lokhov A. Y., Misra S., Coffrin C.: On the Emerging Potential of Quantum Annealing Hardware for Combinatorial Optimization. Journal of Heuristics, (2024, to appear).
- Toth P., Vigo D.: The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. INFORMS J. on Computing **15**(4), 333–346 (2003). `https://doi.org/10.1287/ijoc.15.4.333.24890`
- Voigt(ed.) Der Handlungsreisende wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein ; Mit einem Titelkupf. Voigt, Ilmenau (1832)
- Wolpert D.H., Macready W.G.: No Free Lunch Theorems for Optimization. IEEE Transaction on Evolutionary computation **1**(1), 67–82 (1997). `https://doi.org/10.1109/4235.585893`
- Xavier A.E., Xavier V.L.: Flying Elephants: a General Method for Solving non-differentiable Problems. Journal of Heuristics **22**(4), 649–664 (2016). `https://doi.org/10.1007/s10732-014-9268-8`

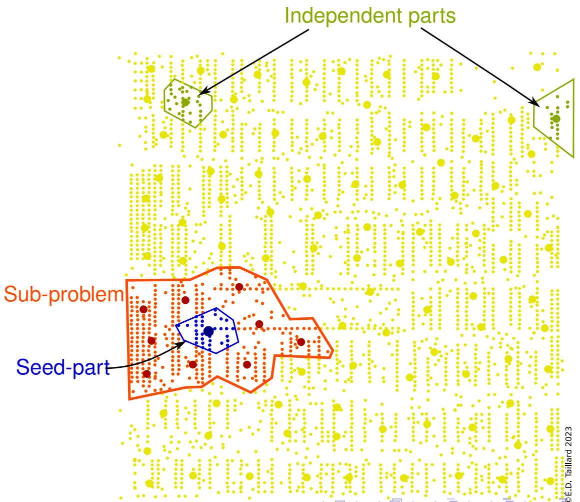
Getting an Appropriate Initial Solution in Linearithmic Time

- Create a tour on a sample of r cities
- Insert the remaining cities in any order, but next to the closest city of the sample
- If the path between 2 cities of the sample has too many cities: decompose it recursively
- Else, optimize the path with a good method (exact, Lin-Kernighan, . . .)



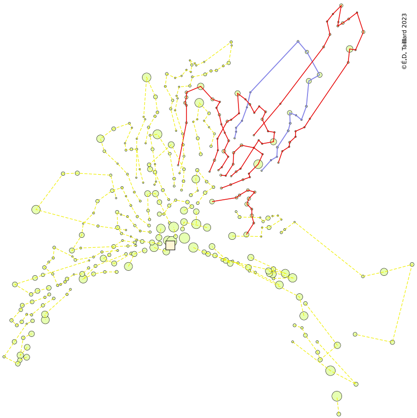
POPMUSIC for Centroid Clustering

- Optimizing 2 cluster well separated cannot improve the solution
- Choose a seed-cluster and the r centres that are the closest
- Optimize these r clusters independently
- Restart with other seed-clusters



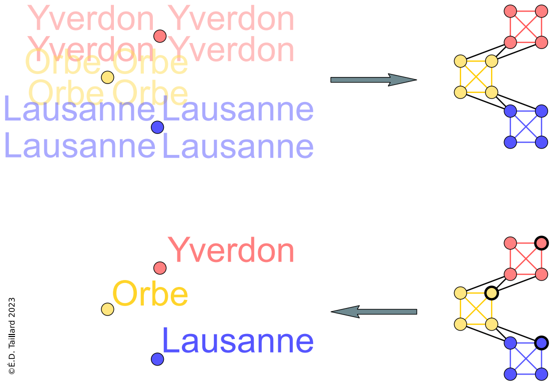
POPMUSIC for the Vehicle Routing Problem

- The customers of a tour is a part
- A subproblem is a VRP with r tours



Map Labelling as a Stable Set Problem

- Create as many node as there are possible label positions for the object
- Connect 2 incompatible nodes by an edge (only 1 label for each object, no overlapping labels)



POPMUSIC Map Labelling

- A part is an object to label (here: 4 possible positions for the label of an object)
- Two objects are at distance 1 if their labels overlap
- Here: 0 is the seed object
- A sub-problem contains 25 objects
  - Labels of red ones taken into consideration but cannot be moved
  - Green ones are ignored