Learning objectives Resources for students Resources for Tutors Tout afficher Tout réduire

TOPIC

Instructions

QUIZ:PROSIT: NO FREE LUNCH

(This prosit will be conducted in the context of the project: if necessary, read the statement again before starting)

That's it, you are finally tackling how to solve your routing problem. Given the constraints and your knowledge of Operations Research, you have decided to focus on metaheuristics. Nothing can stop you now. This time you won't need to call on your new favourite colleague. When suddenly your phone starts vibrating. Of course, it's Agathe who wants to know the latest developments.

After the usual politeness:

Agathe: 'Metaheuristics, great! There are lots of different approaches, what do you have in mind as a solution?'

You: 'I was thinking of starting with a simple heuristic, a greedy one like the Hill Climbing...'

Agathe: 'Oh Okay, I thought you wanted to look into real metaheuristics.'

You: 'But isn't the Hill Climbing one of them?'

Agathe: 'Not really. There is definitely an intensification stage, but no diversification stage at all. With this algorithm, you risk ending up stuck in a local optimum.'

You: 'Well, I'm starting to think about neighbourhood modelling.'

Agathe: 'If you want, but were you planning to start with a trajectory- or population-based metaheuristic? If a genetic algorithm is more suitable, it will help you a lot.'

You: 'See? I would have said that the neighbourhood methods were construction-based methods.'

Agathe: 'Not necessarily. Take the local search methods, for example. They are perturbative. Well, except the Hill Climbing, of course. In fact, there are plenty of ways to categorise metaheuristics. After a while you get lost, and that doesn't help much. But hey, it's always interesting to know the difference between two metaheuristics.'

You: 'In any case, these metaheuristics should all be identical, right? After all, the problem is NP-Complete...'

Agathe: 'Think again, there is no such thing as a free lunch!'

You: 'It's a pity, I was a little hungry...'

Agathe:'... Anyway. You can try a neighbourhood approach if you want. At least you'll be able to test all metaheuristics that work on this method. The tabu algorithm, simulated annealing, GRASP... Nevertheless, take a look at what has already been published on the geometric VRP in scientific journals. Who knows? Maybe there are other metaheuristics that work well. I don't know, genetic or ant colony algorithms, for example. In fact, you should especially see if what matters most for your problem is the intensification or the diversification phase.'

You: 'I guess that depends on how rough the neighbourhood is...'

You: 'Also, if the decision problem is NP-Complete, how can we estimate the quality of a solution?'

Agathe: 'If exact methods were implemented, there may be instances for which the optimal is already known. In the worst case, a lower bound could give you an idea of the quality. It's less accurate than the exact value of the optimal solution, but it's something.'

You: 'Well, I'll try to find that out and I'll run some statistical tests, making the parameters of the algorithm change.'

Agathe: 'Do you already know how to generate your instances? This is important if you want to have representative results. And it would also be interesting to make the parameters of what you generate change.'

You: 'I guess that the larger the network is, the poorer the generated solution will be.'

Agathe: 'That's a possibility, but it's not guaranteed. If I were you, I would think about a design of experiments before running calculations in all directions...'

You: 'All right. And if not, what about the free lunch?'

Agathe: 'I suggest we talk about that again once you have delivered the project!'

Resources for students

Resources for students

- Operations research - An introduction - EN (Chapters 10, 11)
- Introduction to Metaheuristic Algorithms - video EN

EDUCATIONAL GUIDE

Tutoring help

Learning Outcomes

- Determining the characteristics of a hard optimization problem

- Reformulating the hard problem principle using the notions of neighbourhood, greedy algorithm, and local/global optimum

- Listing the main metaheuristics, and explaining how they work

- Differentiating between the diversification and intensification stages

- Solving an optimization problem by means of a metaheuristic

- Breaking down the different steps of a local search method

- Modelling the notion of neighbour of a solution to an algorithmic optimization problem

- Classifying the results from the state of the art with regard to an optimization problem

- Demonstrating the difference between a local optimum and a global optimum

- Creating a program by implementing a metaheuristic

- Using and merging scientific computing libraries

- Building a data structure to represent a problem and its solutions

- Developing a programming code to generate the neighbourhood of a solution

- Developing a metaheuristic by exploiting a neighbourhood generation method convergence

- Comparing the convergence and efficiency of different metaheuristics and different metaheuristic parameters

Prosit philosophy and false leads

- This prosit addresses the processing of **hard optimization problems** using metaheuristics. In the previous prosit, students saw that some optimization problems were hard, and that it was not possible to find an algorithm that would determine an **optimal** solution within a **reasonable** time. This prosit will allow them to better understand the inherent difficulty of these problems through the notions of **local and global optimum**. The neighbourhood approach is emphasised as it makes it easy to understand these notions. Students will then go over the best-known metaheuristics, describing their main characteristics and their overall operation.
- The purpose of this prosit is to solve their problem thanks to a metaheuristic approach (algorithm and data model), based on the state of the art in this field. The 1st workshop is meant to work on implementation skills, as it will allow them to show their ability to implement a method by **Hill Climbing, tabu (or taboo) list, with a multi-start variant**, exploiting the notion of **neighbourhood** of a solution, and to scrutinise its behaviour. That will allow them to find out that implementing a metaheuristic can be quite simple if the tools are suitable (**Python, NumPy, SciPy...**).
- This prosit takes the students to the implementation part of the final project deliverable. A check phase chart is made available to tutors in the project part. This can be used during the Study and Research Activities or at the end of the prosit outcome and feedback to take stock with each project group.

Descriptions of resources

Resources in English:

- Operations research - An introduction - EN (Chapters 10, 11)
- Introduction to Metaheuristic Algorithms - video EN

Original resources in French

- *Scholarvox literature: Précis de recherche opérationnelle, Faure et Robert, Dunod* (introduction and chapter 11)
- *Scholarvox literature: Métaheuristiques, Siarry, Eyrolles* (Foreword)
- *Scholarvox literature: Recherche opérationnelle, Moisdon et Nakhla, Presse des Mines* (chapter 8)

All 3 resources are quite comprehensive. Students can go even further than the specified chapters.

Descriptions of resources

- The notion of greedy algorithm, if not yet fully understood, can be made clearer thanks to this prosit, through the Hill Climbing algorithm implemented in the Workshop.
- One of the risks when studying metaheuristics is to get lost in the classification of the different methods. There are so many different ways of considering the matter that students may end up wasting time. The important thing is not to define the ultimate classification, but to be able to compare two given metaheuristics.
- As mentioned in the 'Prosit philosophy' part, it is necessary to pay special attention to the fact that students must propose a metaheuristic and the related model for the prosit problem.
- Furthermore, they should be able to easily think about the possibility of generating lower bounds using linear programming. The previous Workshop ended with an exercise illustrating this aspect.
- During the prosit kick-off, the tutor can encourage the students to choose a different metaheuristic each, and to present the modelling during the prosit outcome and feedback.
- Some of the resources provided are of a relatively high level. Students are not expected to understand everything, but they must identify the relevant and comprehensible information in these documents.

Prosit Kick-Off

Tutoring help

Discovering and clarifying the situation

- *Unknown words:*

- *Context:*

We are tackling how to solve the routing problem. Given the constraints and our knowledge of Operations Research, we have decided to focus on metaheuristics. But a call from Agathe gives rise to a new reflection.

Analyzing the need

- *Problem:*

Designing a metaheuristic to generate a route while trying to minimize the duration, and which can provide a good quality solution within a reasonable time (a few hours) for the considered problem length.

- *Constraints:*

The computation time must be reasonable

- *Deliverables:*

The implementation method chosen (data model, metaheuristics chosen, settings recommendations for the metaheuristic)

Generalization

- - Specifically solving a hard optimization problem taken from the industry (logistics, production, etc.) using algorithms and Operational Research.

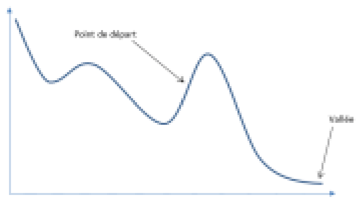
Possible solutions

To guide students in the right direction, a simplified version of the hikers tale can be included as well.

Once upon a time there were 2 hikers lost in the mountain. They must reach the valley, but the fog only lets them observe their immediate surroundings. What should they do when they stumble across a crossroad?

• The first hiker always takes the path that goes down (or the one that goes up the least if none goes down).

• The second hiker follows the path that goes down, but from time to time arbitrarily decides to go up.



Which one can expect to arrive safely at the valley?

What type of algorithm matches with the first hiker's behaviour? Moreover, for which of the two can we reliably foresee what will happen to him? What if the mountain had a different shape? Would the two hikers be lost?

What link can we find with the notions of local and global optimum? Where are they on the diagram? And the neighbourhood?

And with all this fog, how do we know whether we have arrived? If there is a city in the valley, we will know, of course, but if not? What if we had an altimeter that would tell us the height above sea level?

In the previous prosit, we expressed a problem under a generic formulation (a linear program), and we used an algorithm to solve this problem. Could we do the same here, by expressing a problem in a similar form to that of the hiker?

Action plan

1. Listing the most common metaheuristics

2. Selecting a metaheuristic

- Searching the literature for those that have been applied to the VRP (possibly the geometric version)
- Selecting a method that is quick to implement
- Modelling the problem according to this method

3. Implementing the solution
4. Finding a way to assess the quality of the solution

- Searching for already solved instance sets
- Generating instances by making the parameters change
- By lower bounds

5. Analysing the experimental performance of the algorithm

Prosit Outcome and Feedback

Tutoring help

Definitions

- ***Neighbourhood of a solution:***

If we consider a solution to an optimization problem as a configuration, i.e. a set of values defining the decisions corresponding to this solution, a neighbouring solution is one that can be obtained by a sequence of transformations made to the configuration. The neighbourhood of a solution is defined by the total number of neighbouring solutions.

Example: A possible neighbourhood of a spanning tree comprises the total number of spanning trees obtained by modifying only one edge:

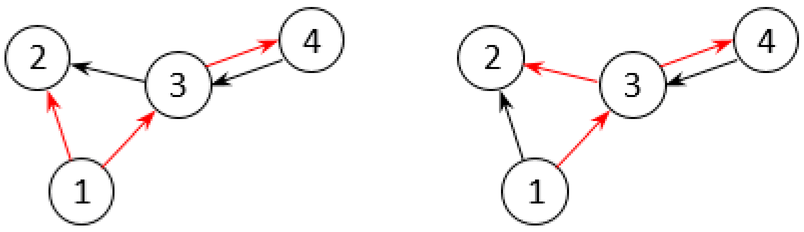


Figure 1 – Deux arbres couvrants voisins enracinés en 1

Figure 1 – Two neighbouring spanning trees rooted in 1

- ***Local and global optimum:***

We consider an optimization problem consisting in minimising an objective function f, and a neighbourhood V.

A global minimum is a solution that is better than (or as good as) any other solution. This corresponds to the bottom of the deepest valley. Formally, a solution s is a global minimum of f if x∈D_f⇒f(x)≥f(s).

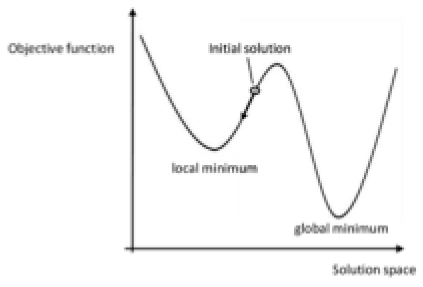


Figure 2 Minimum local et global dans un espace de solutions à 1 dimension

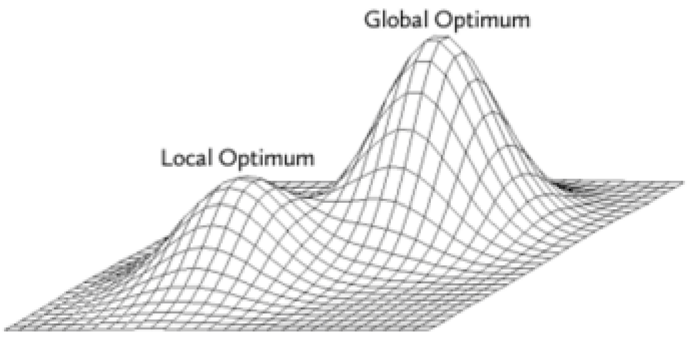


Figure 3 Maximum local et global dans un espace de solutions à 2 dimensions

In the case of a maximisation problem, all we need to do is reverse the inequalities, and talk about a hill rather than a valley.

This definition of a local optimum considers a specific neighbourhood. Students may have come across a variant that considers any possible neighbourhood function (a solution would be a local optimum if there is a neighbourhood for which this solution is better than or as good as all its neighbours).

- ***Local search:***

It is a discrete optimization algorithm that consists of iteratively moving from one solution to a neighbouring solution. Be careful not to confuse this with a local optimization algorithm (typically, the Hill Climbing). The notion of global search is more generic, as it does not necessarily imply selecting the neighbour that has best value (see below).

- ***Greedy algorithm:***

A Greedy Algorithm is an algorithm that makes a local optimum choice on a step-by-step basis. This is sometimes an optimal approach (exact algorithm), such as Prim's algorithm, which builds a minimum-weight spanning tree of a graph by iteratively adding to the tree the smallest value edge that is not part of the tree, until the graph is entirely spanned. However, it is very often a heuristic.

The complexity of a greedy algorithm is often n log(n), but there is a catch: this actually depends on the data structure used to represent the problem (the coding referred to in the NP-Completeness part). Typically, Prim's algorithm has a higher complexity when the graph is represented by an adjacency matrix. This is a reminder of the previous prosit: even though this aspect is usually overlooked, the study of an algorithm's complexity depends on how the data is represented in memory.

- ***Hill Climbing:***

This is a greedy local search algorithm. Local search because it iterates over the neighbours of the current solution, and greedy because it systematically selects the best neighbour. This algorithm merely consists of starting from a solution and systematically selecting the best neighbouring solution until a local optimum is reached (i.e. a solution where all neighbouring solutions are not as good). Like any greedy algorithm, it is a heuristic approach for most problems.

- ***Metaheuristics:***

problems. It is based on the idea that systematically choosing a greedy method for this class of problem can only lock us in a local optimum, and that we must occasionally accept to make a suboptimal choice to escape from this local optimum.

Metaheuristics can be classified according to many criteria, depending on the point of view or the criterion adopted:

- Trajectory (tabu, simulated annealing...) vs Populations (evolutionary...) vs Model (ant colonies...)
- Perturbative (genetic algorithms, local search...) vs Constructive (greedy, ant colonies...)
- Solution-based memory (local search, tabu...) vs model-based memory (estimation of distribution, ant colony...)
- Physical analogy (simulated annealing) or biological analogy (ant colony, genetic algorithm...)
- etc.

The important thing is to note that most metaheuristics always have something in common depending on the perspective from which they are considered.

A particular class has emerged over the last twenty years: hybrid methods, which combine several approaches. For example, the memetic algorithm extends evolutionary algorithms by hybridising with a local search algorithm.

- Intensification, diversification:

Since these notions are quite abstract, students are not required to fully master them.

Intensification consists of concentrating the search on a subset of solutions (or neighbours of the current solution), which we know from experience that tends to produce good solutions. Conversely, diversification consists of moving away from the current solution, so as to reach new areas that potentially hold better combinations.

When applied to local search methods, this often translates into:

- Intensifying by favouring better quality neighbours
- Diversifying is basically authorising (with a fairly low probability) the selection of poorer quality neighbours.

For constructive methods, this translates into:

- Intensifying by favouring components that have belonged to the best combinations already built.
- Diversifying is basically choosing (with a fairly low probability) components from combinations that are not as good

Intensifying makes it possible to converge faster, but at the expense of a greater risk of getting stuck in a local optimum. Conversely, diversifying increases the probability of finding a global optimum, but at the expense of an increasing convergence time.

The balance between intensification and diversification depends on the problem considered, but also (and above all) on the computation time we can afford.

- **Tabu (or taboo)**

Metaheuristic that performs a greedy local search, while forbidding to visit solutions that have already been explored recently (the so-called tabu/taboo solutions). As a result, if all better-quality neighbours are tabu, the algorithm will select a poorer solution, and potentially escape a local optimum. The implementation is usually done by way of a tabu list, and the tabu method's behaviour mainly depends on this list (an increase in its length represents diversification, a reduction corresponds to intensification).

The length can be fixed, depending on a parameter of the problem (typically its length), or change (randomly or not). In addition, there are many variants. We can diversify by occasionally authorising unfeasible solutions in order to escape a local optimum (ejection chain method). We can intensify by restricting the neighbourhood to a subset selected according to criteria deemed to favour the value of the solutions, or we can decide to return to the best solution found and resume the search by intensifying it (reduction of the tabu list length, for example).

- **Simulated annealing**

Metaheuristic inspired by annealing methods. Annealing is a method for treating a material in order to alter its state (magnetic properties, for example), by means of successive heating and slow controlled cooling cycles. The simulated annealing metaheuristic is based on this analogy, by considering notions of temperature and energy level of the system (representing the value of the current solution).

- **GRASP**

Metaheuristic that performs a local search (improvement phase) by iteratively considering several randomly generated initial solutions (construction phase). The improvement phase can be carried out with any local search method (tabu, simulated annealing, Hill Climbing), or even by combining several methods (tabu then Hill Climbing, or vice versa, Hill Climbing followed by tabu...). The construction phase is more challenging because it requires approaches that are sometimes quite sophisticated (greedy randomised by a candidate list, problem-specific heuristic...).

- **Guided local search**

A local search method (often a simple local optimization) in which we make the objective function change so as to render the local optima already visited less attractive (for example by adding to their objective value a bonus weighted by a weight that changes according to the number of visits already made). This approach can be regarded as a variant of the tabu, except that the solutions already visited are no longer simply forbidden, but simply discouraged. Moreover, the tabu method uses a short-term memory, while the guided local search uses a long-term memory (indirectly, through the weights of each solution).

- **Variable neighbourhood search**

This is a local search method that considers several notions of neighbourhood at the same time. At each iteration, we select the best neighbouring solution of the current solution in all the considered neighbourhood types.

- **Evolutionary algorithms**

These are constructive, distributed, population-based metaheuristics, also called genetic algorithms, which are explicitly inspired by Darwin's theory of evolution. We consider a set of individuals that form a population of generally constant length, where each individual is a possible solution to the problem. One iteration of the algorithm corresponds to one generation. At each generation, individuals appear or disappear due to the successive application of operators:

1. Parental/parent selection: selection of parents to father children
2. Crossover and mutation of selected parents
3. Children's performance assessment
4. Environmental selection among all (already existing and newly created) individuals to form the next generation

These operators are divided into two categories: (parental and environmental) selection operators, and (mutation and crossover) variation operators from among existing individuals.

An individual's ability to be selected depends on their performance. The variation can be completely random (and allow extensive exploration, but poor convergence time), or completely deterministic and greedy (a Hill Climbing, which might lock the global algorithm in a local optimum). The algorithm's efficiency depends on the right balance between the impacts caused by selection and variation operators. The link with intensification and diversification is obvious.

- **Colony of artificial ants**

(this is one of the candidates that students can propose). The general principle is the following:

- All ants travel one of the possible paths

- At each stage, an ant chooses to move from one city to another by complying with the following rules:

- that ant can only visit each city once
- the further away a city is, the less likely it is to be chosen
- the greater the intensity of the pheromone trail on the edge between two cities, the more likely it is that the path will be chosen
- once its path is completed, the ant leaves more pheromones on all the edges it has travelled along if the path is short;
- the pheromone trails evaporate with each iteration.

This approach can be generalised to any type of graph problem:

- Initialisation of pheromone trails
- Looping as long as the stop criterion is not met:
- Building solutions on a component-by-component basis
- Updating the pheromone trails

The generalisation to problems other than graph problems is rather difficult, as the modelling is highly dependent on the problem considered.

- ***No Free Lunch Theorem***

This theorem's mathematical fundamentals are quite difficult to understand, but the actual impact is that, when considering NP-Hard problems, no metaheuristic can have efficient results in all cases. There will inevitably be better methods on some problems, which will be less efficient on others.
Validation questions

- What is a local search? Is it a metaheuristic?

A local search uses the notion of neighbourhood to move from one solution to a neighbouring solution. It is not necessarily a metaheuristic. Hill Climbing is a local search that does not include any diversification. Therefore, it is a greedy method that calculates a local optimum.

- What are the two main types of operations that allow us to guide a metaheuristic? What do they comprise, what are they for?

Intensification and diversification. Intensification tends to reinforce the search for a local optimum, diversification tends to move away from the valley we are in.

- Which notion is exploited by Tabu, Simulated Annealing, GRASP, and Hill Climbing? How? What determines its efficiency?

The notion of neighbourhood. Two solutions are neighbours if we can move from one to the other by means of a simple transformation. This allows us to design algorithms that iterate and explore the landscape of feasible solutions in an attempt to find the optimal solution. These algorithms work best when the landscape drawn by the solutions is not too steep (landscape roughness).

- Generally speaking, which metaheuristic is the most efficient?

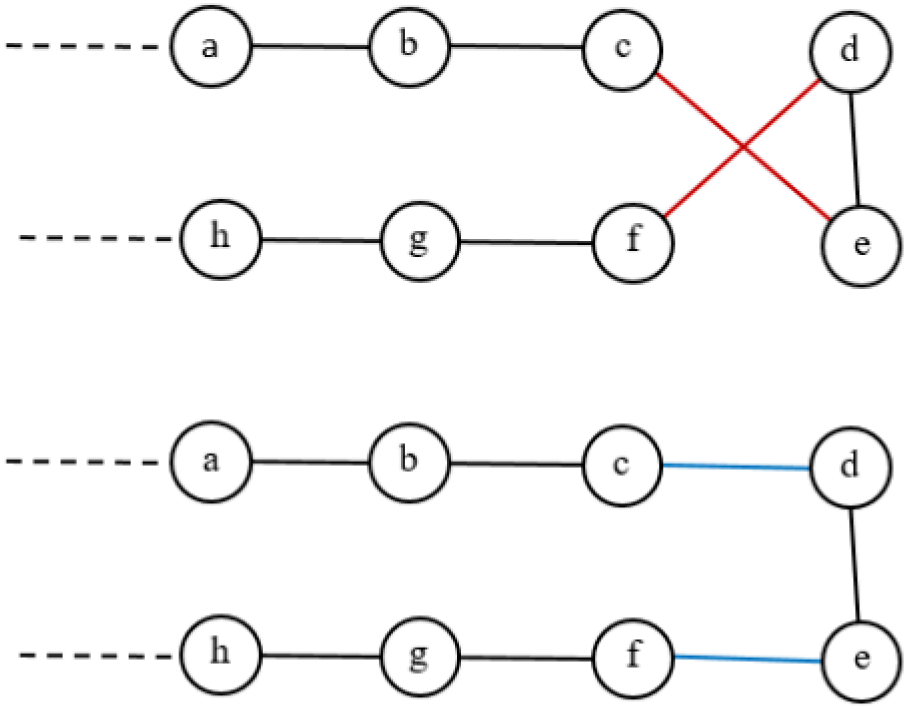
Reminder: following the previous prosit, the problem considered is the geometric edge-weighted travelling salesman problem. In fact, we consider a non-complete graph, which we complete with the missing edges, weighted with the length of the shortest path between both ends (see solution of the previous prosit).

There are many acceptable solutions, all metaheuristics can work. Below are some valid proposals. In any case:

- A solution is a sequence of vertices
- A valid solution is a cycle that passes at least once through each vertex
- The value of a solution is the sum of the weights of its edges.

Neighbourhood methods (Tabu, simulated annealing, GRASP, variable neighbourhood...)

The neighbourhood of a solution consists of all the solutions obtained by removing two edges and then reconnecting the two paths obtained with two other edges (a move called 2-opt). The two edges to be added must therefore be chosen from all those reconnecting the two paths.



Les arêtes en rouges ont été supprimées et remplacées par les arêtes en bleu

The red edges have been removed and replaced by the blue edges

Note: We can also consider the variants 3-opt, 4-opt, etc.

It is easy to demonstrate that this definition allows us to obtain a neighbourhood where all solutions are feasible.

Another approach is to considerer swaps of vertices instead of edges. This variant is particularly found when using simulated annealing.

Genetic algorithm

For example, for the two-point crossover method:

The genes between the r and s positions of one parent are exchanged with those of the other parent to generate offspring. We add, to the child, the vertices between r and s of the first parent at positions r, r+1,..., s and then we complete the positions before r and after s with the vertices of the second parent which are not in the solution yet (in order of appearance in the second parent).

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

×

9	8	7	6	5	4	3	2	1

=

9	5	4	3	2	6	7	8	1

We generate two children according to the order in which we have selected their parents. We can then add the best of the two children (or both) to the new population. This mutation is repeated until the required number of children is reached.

Mutations may involve reversing two randomly selected genes in the solution.

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

1	2	**8**	4	5	6	7	**2**	9

This method can be tested on known instance sets, such as:

http://www.vrp-rep.org/faq.html

Or be assessed against random instances on which lower bounds are calculated. Typically, a linear program allows us to obtain an unfeasible solution that can be used to assess the quality of a solution. It is not necessary to provide the entire linear program, but just in case, here it is:

We associate a distance δi,j equal to di,j with each pair (i,j) of cities to visit (i ≠ j) if there is a way to go directly from i to j (i.e., (i,j) ∈ V(G)), otherwise we set the distance to 1, and a binary sequence variable, xi,j , which takes the value 1 if city j is visited immediately after city i in the route, taking the value 0 otherwise. Thus, the TSP is modelled by:

$$
\begin{array}{lll}
Min & \sum_{i=1}^{n}\sum_{j=1}^{n} \delta_{i,j}.x_{i,j} & \\
\text{s.c.} & \sum_{j=1}^{n} x_{i,j} = 1 & \forall i = 1..n \\
 & \sum_{i=1}^{n} x_{i,j} = 1 & \forall j = 1..n \\
 & \sum_{i \in S, j \notin S} x_{i,j} \geq 2 & \forall S \subset X, S \neq \emptyset \\
 & x_{i,j} \in \{0,1\} & \forall i = 1..n, \forall j = 1..n
\end{array}
$$

The first two constraints reflect the fact that each city must be visited exactly once; the third constraint rules out solutions composed of disjoint subtours, which is usually called subtour elimination constraint.

TOPIC

Instructions

QUIZ

Workshop Statement [zip] (Notebook Jupyter)

EDUCATIONAL GUIDE

Standard corrections

Workshop version for tutors [zip]

WORKSHOP 2

TOPIC

Instructions

QUIZ

Workshop Statement [zip] (Notebook Jupyter)

EDUCATIONAL GUIDE

Standard corrections

Workshop version for tutors [zip]