

Learning
objectivesResources for
studentsResources for
TutorsTout
afficherTout
réduire **TOPIC** **Instructions****QUIZ:PROSIT: NO COMPLEX**

Your collaboration with Agathe has been very rewarding. But now that your colleague is going to continue her project together with the Grand Est region, it's time for you to fully engage in answering ADEME's call.

The main advantage is that you will be able to directly use this work in the management of delivery routes. All you have to do is replace the bridges with cities. But when it's time to wish you all the best, Agathe, who has overheard you, can't help but intervene:

Agathe: 'It's not that simple, the problem is not the same...'

Your enthusiasm fades instantly. Agathe then proposes to stay with you a little longer to think things through.

Agathe: 'Do you think that passing through each city only once is the same as passing through each path only once?' Moreover, it's not a question of knowing whether it is possible to pass through once, but to do the route no matter what, while minimising the time it takes. It's an optimization problem.'

You: 'We could try to sort the cities by travel time required going from the starting point, and choose... Or we calculate the shortest paths between each pair of cities...'

Agathe: 'Not so fast! You're already trying to find an algorithm, while you have not yet modelled the decision problem corresponding to this optimization problem!'

You: 'What's the purpose of the decision problem? All we want is to find the best solution!'

Agathe: 'Depending on the complexity class of the decision problem associated with the optimization problem, the methods you're envisioning will most likely be heuristic, or of exponential complexity. This complexity must be established in order to know if we can look for an optimal algorithm that might stand a chance of being within a reasonable asymptotic complexity class. This is crucial, otherwise you risk spending the next few weeks coding for nothing.'

You: 'When we launched the project, I heard others say that it was an application of the travelling salesman problem. Is its complexity known? We'll have the answer straight away.'

After a short pause to think...

Agathe: 'It makes sense, but it seems to me that the travelling salesman problem has different constraints on the graphs compared to your case. Besides, you absolutely need a route as output. I know there's a metric version of the problem. Perhaps this version fits better here... You should be able to find that in the literature, as there must be dozens of scientific articles that address the subject.'

In any case, if you find that you can start by using this problem and it's already proven to be NP-Complete, the formal proof by polynomial-time reduction should be easy to get.'

You: 'And if we actually find an article proving that it is NP-Complete, what do we do? Do we start with a heuristic algorithm?'

Agathe: 'Start by determining the time complexity of your problem. Or better yet, first check if the associated decision problem is in NP. Imagine if a Turing Machine is not even capable of deciding on the correctness of a solution, not even

Advanced Algorithms **Preparation** **Project** **1 - PBL Loop** **2 - PBL Loop** **3 - PBL Loop** **4 - PBL Loop**

You: 'And the space complexity too, right?'

Agathe: 'In essence, you're right to mention that, but given the capabilities of modern computing systems, it shouldn't be a problem.'

What is clear in all this is that the code is not needed straight away.

▼ Resources for students

- P NP and NP Completeness :The Basics of Computational Complexity [🔗](#) (chapters 2, 3 and 4) - EN
- P, NP, NP-Hard & NP-complete problems [🔗](#) - EN
- NP-Hard and NP-Complete Problems [🔗](#) video - EN
- Computers and Intractability - A Guide to the Theory of NP Completeness [🔗](#) - EN
- P NP NP-Hard NP-Complete||Design and Analysis of Algorithm [🔗](#) video - EN
- What is the Traveling Salesman Problem? [🔗](#) - video EN
- Complexité algorithmique partie 2 - NP-Complétude [🔗](#), Benjamin COHEN BOULAKIA (2020) - FR continuation of the video from Prosit 1

EDUCATIONAL GUIDE

▼ Tutoring help

Learning Outcomes

- Defining the right approach to solve an optimization problem
 - Arranging the different complexity classes in a specific order
 - Interpreting the complexity of an algorithm or problem from an implementation perspective
- Studying the complexity of a decision problem
 - Determining whether a decision problem is in NP by performing a polynomial test on a candidate solution
 - Determining a reasonable time complexity according to the context
- Estimating the difficulty of an optimization problem
 - Linking an optimization problem to a decision problem
 - Classifying an optimization problem using a complexity study of the associated decision problem
- Using the theoretical notions of algorithmic complexity
 - Distinguishing between P class and NP class
 - Reformulating the position of an NP-Complete problem in NP

Advanced Algorithms **Preparation** **Project** **1 - PBL Loop** **2 - PBL Loop** **3 - PBL Loop** **4 - PBL Loop**

- Classifying the orders of magnitude of algorithmic complexity
- Placing algorithmic complexity questions in the context of the algebraic language theory and the computability theory
 - Expressing the fact that the notion of complexity depends on the coding of an instance
 - Stating the notion of strong NP-Completeness that results from it
 - Mentioning the link between the complexity of a decision problem and the Turing machine

Prosit philosophy and false leads

- This prosit establishes the link between the decision problem and the optimization problem. It leads students to formalize the notion of algorithmic complexity, to deal with NP-Completeness proofs, and to draw the necessary conclusions regarding the specific optimization problem they have to address.
- Operational skills include the ability to assess the complexity of an algorithm, and to determine the link to decision problems. The goal is not to be able to find an NP-Completeness proof of a decision problem by polynomial-time reduction, but to understand and restore the meaning of a proof already available in the literature.

On the other hand, students must be able to determine the impact of such a result on an associated optimization problem and, in general, to understand the impact of a theoretical complexity on the actual use of an algorithm.

- This prosit must lead to the execution of the formal project modeling deliverable.

Descriptions of resources

Resources in English

- P NP and NP Completeness :The Basics of Computational Complexity [📄](#) (chapters 2, 3 and 4) - EN
- P, NP, NP-Hard & NP-complete problems [📄](#) - EN
- NP-Hard and NP-Complete Problems [📄](#) video - EN
- Computers and Intractability - A Guide to the Theory of NP Completeness [📄](#) - EN
- P NP NP-Hard NP-Complete||Design and Analysis of Algorithm [📄](#) video - EN
- What is the Traveling Salesman Problem? [📄](#) - video EN
- Complexité algorithmique partie 2 - NP-Complétude [📄](#), Benjamin COHEN BOULAKIA (2020) - FR continuation of the video from Prosit 1

Original Resources in French

Scholarvox literature: Algorithmes, Cormen, Dunod [📄](#) : complexité (chapter 10)

Literature on the entire part concerning algorithms. The specified chapter addresses complexity

- *Scholarvox literature: Optimization combinatoire, Sakarovitch, Herm* [📄](#) *ann* : de l'efficacité des algorithmes à la complexité des problèmes (chapter 2)

The specified chapter addresses the efficiency of algorithms and their complexity

- *Scholarvox literature: A la découverte des Graphes et des algorithmes de graphes, Chris* [📄](#) *tian Laforest, EDP Sciences* (chapters 14 to 18)

Book on graphs. Chapter 14 addresses hard problems. Chapter 18 addresses the travelling salesman problem.

- *Complexity of algorithms: P Versus NP (video)* [📄](#)


Explanatory video on the P=NP problem

- *TSP: Travelling Salesman Problem (video)* [📄](#)

Explanatory video on the travelling salesman problem

- *Algorithmic complexity - Benjamin COHEN BOULAKIA* [📄](#)

Advanced Algorithms **Preparation** **Project** **1 - PBL Loop** **2 - PBL Loop** **3 - PBL Loop** **4 - PBL Loop**

Ouvrage de référence Scholarvox : A la découverte des Graphes et des algorithmes de graphes, Christian Laforest, EDP Sciences  (chapitres 14 et 18) - FR

Notes for the tutor, tricks and tips

- Some of the underlying notions are quite complex, and it is important that students do not waste too much time trying to deeply understand these notions, which are too advanced for them. This is the case for all notions corresponding to the educational goals of level 1 – Knowledge, and to a lesser extent those of level 2 – Understanding. The tutor should therefore ensure that students do not set excessively high goals on these matters. Moreover, many of the resources provided go quite far into theory, so students need to know how far to go when reading a given resource.
- Generally speaking, this is a very theoretical prosit, there are no programming exercises, and the exercise series takes the notions through demonstration and interpretation exercises. Students who are most put off by scientific computing and theory are likely to drop out, or even fail to grasp the usefulness of this prosit. It is important to keep reminding students that this analytical work is necessary and inherent to any industrial project, particularly in terms of logistics.
- Prosit's goal is not to solve the Hamiltonian cycle problem (which they should immediately identify, since it has already been addressed in the previous prosit), either in an optimal or approximate way (the dialogue between Agathe and the interns should make students understand this), but only to determine whether or not it is reasonable to envisage an optimal algorithm. The final deliverable is the 'no' answer substantiated by an NP-Completeness proof (membership in NP and polynomial-time reduction). Care must be taken to ensure that students do not engage in actually solving the problem.

✓ **Prosit Kick-Off**

✓ **Tutoring help**

Discovering and clarifying the situation

- *Unknown words:*

NP-Complete, Heuristic, Asymptotic complexity class, Certificate algorithm, Polynomial-time reduction, P, NP, Turing machine, Temporal complexity/Time complexity, Spatial complexity/Space complexity, optimization problem

- *Context:*

After having worked on routes to carry out the street light maintenance, we are now focusing on the project to answer ADEME's call and the management of delivery routes.

Analyzing the need

- *Problem:*

This is not the same kind of problem and we cannot apply the same algorithm. We do not even know if it can be solved in an optimal way within a reasonable time.

- *Constraints:*

The answer to the question must be formally proved.

- *Deliverables:*

The decision problem associated with the maintenance route optimization problem, and a proof of its complexity.

Generalization

- Using theoretical tools to determine the best way to achieve a specific goal.

Possible solutions

- Students should quickly identify the link between the route problem considered here and the Hamiltonian cycle of minimum length.

Advanced Algorithms **Preparation** **Project** **1 - PBL Loop** **2 - PBL Loop** **3 - PBL Loop** **4 - PBL Loop**

- They have to guess that 'outside of NP' is very bad news, and that 'in P' is good news. There is no indication regarding the meaning of 'NP-Complete', but they can assume that this is an intermediate case. They must also understand that the NP-Completeness proof is a polynomial-time reduction (without knowing what it corresponds to). As for the proof of membership in NP, the method is stated almost explicitly.
- It is necessary to make students think about which complexity seems the most suitable. The exponential complexity mentioned in the prosit is a line of thought. Moreover, time complexity is the only complexity considered here, space complexity is not addressed at all. Normally students should ignore this notion, something that is explicitly stated in the prosit.
- Finally, the text makes it quite clear that proofs should not be determined by the students themselves, but found in the literature. Nevertheless, they must be aware of the need to understand these proofs.

Action plan

1. Modelling the route problem
 - Searching for the definition of the metric TSP problem
 - If possible, modelling the route problem as an optimization problem of the metric TSP
 - Otherwise, finding another optimization problem for modelling the route problem
 - Determining the associated decision problem
2. Determining the complexity of the decision problem
 - Finding a certificate algorithm proving the membership of the Hamiltonian cycle of minimum length in NP
 - Searching the literature for a polynomial-time reduction proving NP-Completeness
3. Drawing a conclusion on the asymptotic complexity of the optimization problem

✓ Prosit Outcome and Feedback

✓ Tutoring help

Definitions

- **Algorithmic problem:**

It is possible that students did not see fit to formalise the notion of problem, either in this prosit or in the previous one, but this definition is necessary if we are to be able to simply talk about a problem instance.

A problem is:

- A data input format that is usually binary-coded (sometimes unary-coded for pseudo-polynomial problems), thus defining a set of definitions for each piece of data
- A question about this data

A instance is a set of values that complies with the data format (for example, for the Eulerian cycle problem, it is a graph with a starting vertex).

- **Decision problem:**

Decision: Algorithmic problem whose question accepts either 'yes' or 'no' as an answer. Optimization problem associated with a decision problem:

Advanced Algorithms Preparation Project 1 - PBL Loop 2 - PBL Loop 3 - PBL Loop 4 - PBL Loop

associating an optimization problem with a decision problem. Thus, solving an optimization problem is basically finding the smallest or the largest solution for which the answer to the decision problem is 'yes'.

Example:

The shortest path in a graph.

The decision problem is expressed as follows:

Given a graph G , two vertices u and v of G , and a integer k , is there a path from u to v , of lesser than or equal length to k ?

The optimization problem is expressed as follows:

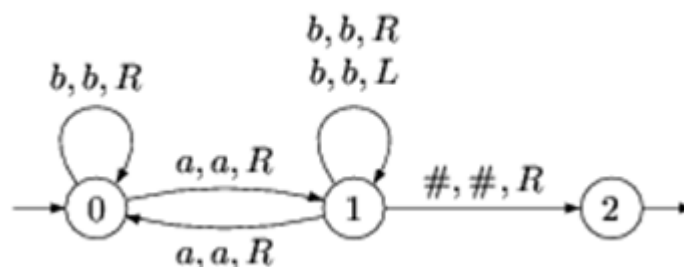
What is the smallest value of k for which the answer is 'yes'?

- **Turing machine:**

Mathematical model proposed by Alan Turing of how a computing system works. A Turing machine is defined by

- A finite set of possible states
- An alphabet of symbols including the blank symbol
- A transition function, associating a triplet (state, symbol, motion direction) with each pair (state, symbol)
- The set of accepting states, subset of possible states
- An initial state

Example of a Turing machine:



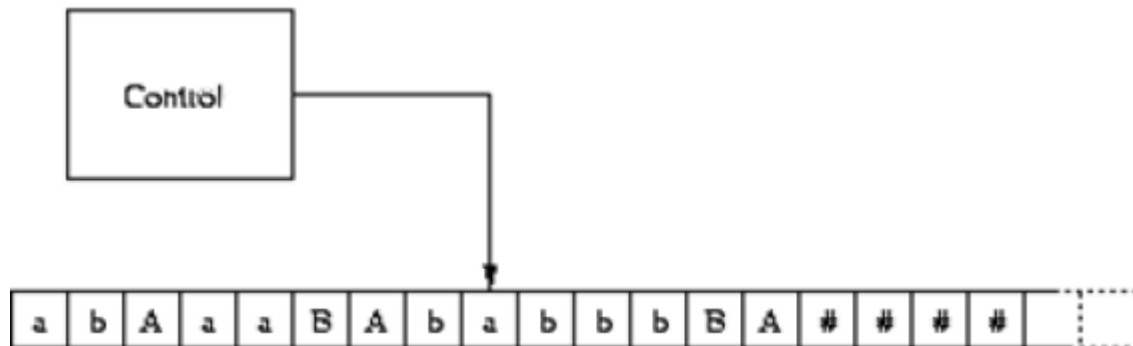
A Turing machine is equipped with a tape that can contain symbols from the alphabet, and a read/write head located on a given box of the tape. At each step, the machine is in a state and reads the symbol x_n on the current box of the tape. The corresponding transition function (e_{n+1}, x_{n+1}, s) is applied as follows:

- It changes to the new state e_{n+1}

- It writes the symbol x_{n+1} on the tape

When the machine detects a final state, it stops.

Example of a Turing machine using a tape:



This theoretical machine is used to model a computing system capable of processing decision problems: A Turing machine capable of solving a decision problem receives the instance encoded in the machine's alphabet on its tape, and applies the transitions until it stops. If the machine stops on an accepting state, the answer to the instance is 'yes', otherwise the answer is 'no'. As a result, the entire complexity theory based on the Turing model is about decision problems.

Note: once this definition is provided, we notice that the algorithmic complexity according to the Turing model is not expressed according to the amount of data, but according to the length of its coding. The coding length issue is usually discarded depending on the data length. Typically, in the case of a graph problem, we usually reason about the number of vertices or edges of its sagittal representation, whereas the complexity of an algorithm essentially depends on the length of its representation in the alphabet used (usually binary): adjacency list, matrix... In some cases (beyond the scope of this prosit), this shortcut cannot be made, as the issue is markedly more complex (especially for pseudo-polynomial algorithms).

- **P class:**

An algorithm of complexity f is in the P class (it is said to be polynomial) if there is a polynomial p such that f is in $O(p(n))$. This class contains all the so-called easy problems in the algorithmic sense.

The term 'P' simply means 'Polynomial'.

- **NP class and certificate:**

A decision problem is in NP if there is an algorithm that uses, as an input parameter, an instance of the problem, a candidate solution, and that checks in polynomial time whether the candidate solution makes it possible to answer 'yes' to the instance of the problem.

Example:

- Is a 3-coloring graph (can its vertices be coloured with only 3 colours, so that no neighbouring vertices have the same colour)?
- A certificate is an algorithm that uses a graph and a colouring as parameters and checks, in polynomial time, if the colouring is valid (does the colouring comply with the adjacency constraint? Does it use at most 3 colours?)

Advanced Algorithms Preparation Project 1 - PBL Loop 2 - PBL Loop 3 - PBL Loop 4 - PBL Loop

Another definition involves the notion of non-deterministic Turing machine. This is what the term NP means. Non-deterministic Polynomial (and not 'Non-Polynomial').

It is a Turing machine in which some transitions are non-deterministic (there are several possible ending states). It is not necessary to fully master this approach. However, intuitively, we can guess that the comprehensive exponential complexity approach is behind it, since the only way we know to check an instance with a non-deterministic Turing machine is to test all the combinations of non-deterministic transitions that we detect when reading the instance.

We easily demonstrate that $P \subseteq NP$.

- **Classe co-NP :**

Let P be a decision problem. The complement problem of P , denoted \bar{P} is the decision problem such that $p \in O(P) \Leftrightarrow p \notin O(\bar{P})$

Definition: co-NP is the set of all decision problems P such that $\bar{P} \in NP$

We easily demonstrate that $P \subseteq \text{co-NP}$.

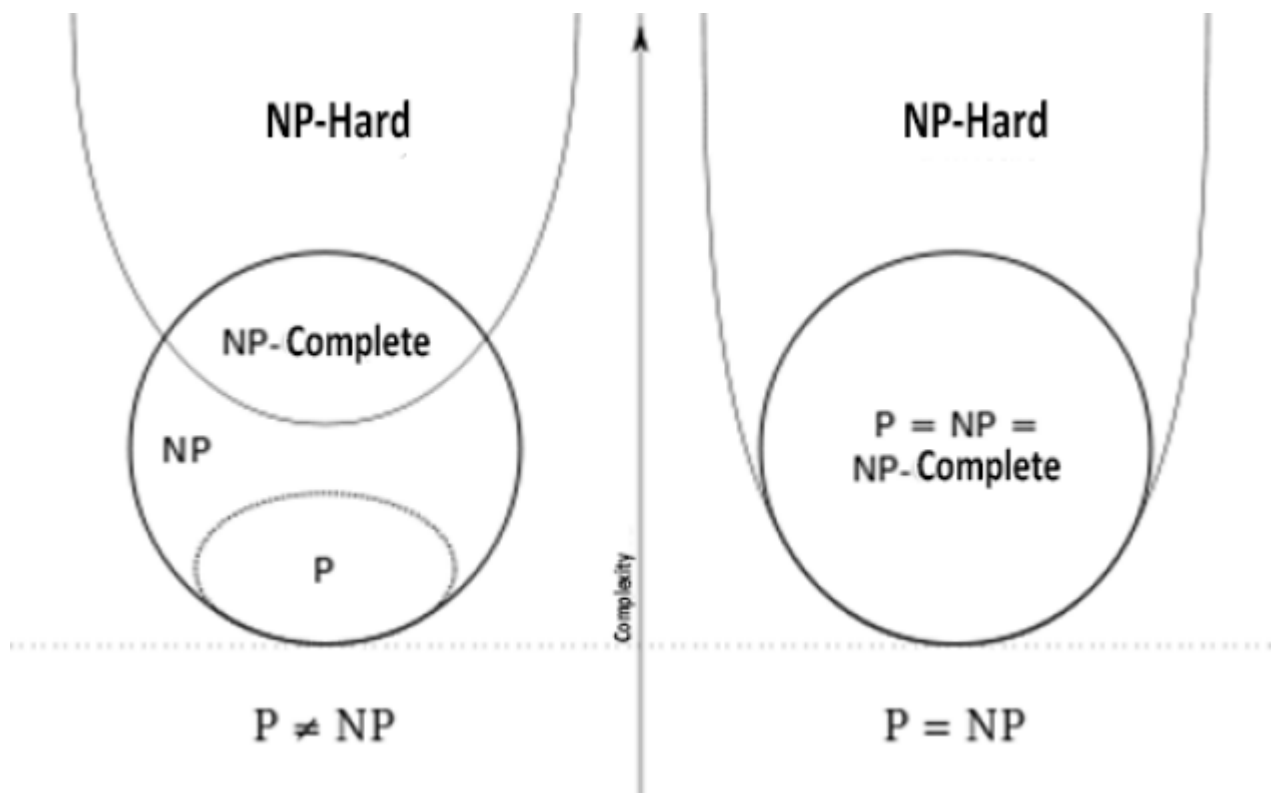
- **NP-Hard problem, NP-Complete problem:**

An NP-hard problem is a problem to which any problem of class NP can be reduced by a polynomial-time reduction (definition further below). In other words, a problem is NP-Hard if it is at least as hard as any problem in NP. Note that an NP-Hard problem is not necessarily a decision problem. Intuitively, it makes sense that optimization problems are at least as hard as the decision problems on which they are based.

An NP-Complete problem is an NP-Hard problem that belongs to NP (therefore a decision problem). The class of NP-Complete problems is therefore the class of the hardest problems in NP. Proving that a problem is NP-Complete is basically proving that it is in NP, and that it is at least as hard as any other NP-Complete problem. This second step is the polynomial-time reduction operation, defined below.

This definition introduces one of the speculations for which the Clay Mathematics Institute is offering a million-dollar reward (Millennium Prize Problems): is $P=NP$ or $P \neq NP$?

Complexity classes, according to the two possible hypotheses:



- **Polynomial-time reduction:**

A reduction is a way of reducing the solution process of one problem to that of another. We consider reductions that take into account the computation time of the function that makes the transition from one problem to another. Applied to decision problems, a polynomial-time reduction matches an instance of the problem P_2 with an instance of the problem P_1 which has the same answer and is computable in polynomial time (i.e., the transformation of the instance of P_1 into an instance of P_2 is done in polynomial time according to the coding length of P_1). The algorithm performing the transformation is known as a reduction algorithm. We denote it as $P_1 \leq P_2$ when there is such a polynomial-time reduction from problem P_1 to problem P_2 . We then say that P_1 is reducible in polynomial time to P_2 .

In a more general case where we consider any type of problem (especially optimization problems), polynomial reduction includes a third step: the transformation of an admissible solution (optimal in the case of optimization problems) of P_2 into an admissible solution of P_1 , a transformation that must be done in polynomial time.

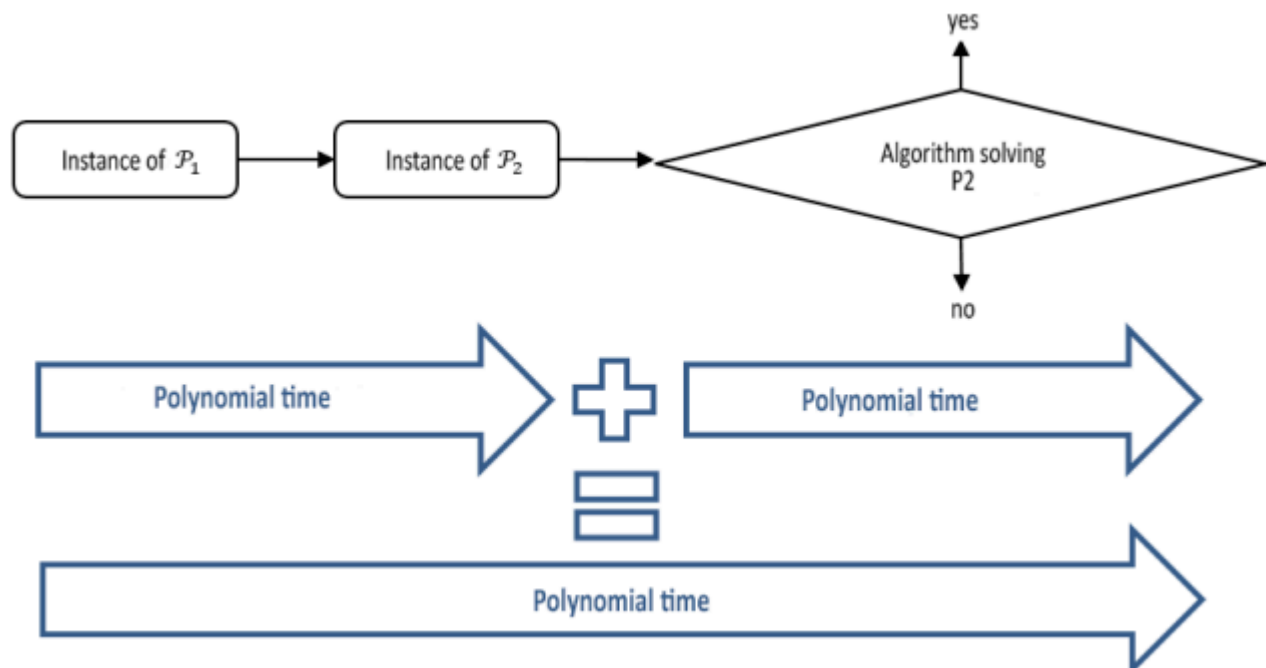
It is easy to demonstrate that a polynomial-time reduction of P_1 to P_2 proves that P_2 is at least as hard as P_1 . To solve P_1 , one just has to transform an instance of P_1 into an instance of P_2 then solve this instance, all in polynomial time, and transform the solution of P_2 into a solution of P_1 . If we know how to solve P_2 , we know how to solve P_1 . Intuitively, he who can do more can do less.

More formally:

If P_2 is a problem that can be solved in polynomial time and if P_1 is reduced in polynomial time to P_2 , then P_1 can be solved in polynomial time.

If P_1 is a problem that cannot be solved in polynomial time and if P_1 is reduced in polynomial time to P_2 , then P_2 cannot be solved in polynomial time either.

Polynomial-time reduction proving that P_2 is at least as difficult as P_1 :



It should also be noted that the relation 'is reducible in polynomial time to' is a transitive relation: $P_1 \leq P_2$ and $P_2 \leq P_3 \Rightarrow P_1 \leq P_3$.

Following a similar reasoning, we can also postulate that if a decision problem is NP-Complete, then its associated optimization problems cannot be processed in polynomial time. Conversely, if an optimization problem admits a polynomial algorithm, its associated decision problem is in P. Sometimes we find the NPO-Complete notation, but be careful because this notation is used in a special context with specific theoretical needs (approximation algorithms).

- **Cook's theorem:**

Advanced Algorithms Preparation Project 1 - PBL Loop 2 - PBL Loop 3 - PBL Loop 4 - PBL Loop

Original NP-Complete problem is the SAT problem, the satisfiability problem of a logical formula. Any NP problem can be formulated as a SAT problem.

- **Travelling Salesman Problem:**

Also called the VRP problem, the Travelling Salesman Problem, or TSP, corresponds to the problem of a travelling salesman who has to visit all the cities in his network, and wishes to travel the shortest distance possible. It is expressed as follows:

Data	$G=(V, E, P)$ a complete graph with weighted edges $v \in V$
Objective	a Hamiltonian cycle of minimal size starting from v

The associated decision problem is:

Data	$G=(V, E)$ a complete graph with weighted edges $v \in V$ $k \in \mathbb{Z}$
Question	Is there a Hamiltonian cycle starting from v , of lesser or equal size to k ?

The optimization problem is then formulated as 'What is the smallest value of k for which the answer is yes?'. It is NP-Complete:

- Membership in NP

The certificate algorithm is very simple, since it uses, as an input parameter, a sequence of vertices and checks:

- That it is indeed a cycle starting from v (trivial)
- That its length is less than or equal to k (trivial)
- That it is indeed Hamiltonian (all we have to do is mark the vertices along the way and check that a vertex is not marked twice)

Its complexity is linear (therefore, a polynomial complexity of order 1). From this, we draw the conclusion that the Hamiltonian cycle problem is in NP.

- NP-Completeness

Here, the proof is quite simple, as we just have to go back to the Hamiltonian cycle problem (without values on the edges). This problem has been known as NP-Complete since 1972.

- We consider an instance of the Hamiltonian cycle problem, i.e. a graph $G=(V, E)$, and then we build an instance of the travelling salesman:

o Let $G' = (V, E', P)$ be the complete edge-weighted graph built on the set of vertices V , with distances $P(e)$
 $=1$ if $e \in E$ and $P(e)=2$ otherwise.

o We take $k=|V|$

- **Metric version of the Travelling Salesman Problem**

Known in English as Δ -TSP, this is a special case in which we consider a graph whose distances comply with the triangle inequality: for all vertices i, j , and k , $P(i-j) \leq P(i-k) + P(k-j)$.

It is easy to demonstrate that this special case remains NP-Complete, since the demonstration used for the general TSP builds graphs with lengths 1 and 2 that comply with this very inequality. An algorithm capable of solving the TSP in metric version only would, nevertheless, also be capable of solving the Hamiltonian cycle problem.

On the other hand, some heuristic algorithms tend to behave better on metric cases than in the general case.

Validation questions

- If P means 'polynomial', what does NP mean?

'Non-deterministic Polynomial' (and not 'Non-Polynomial'), which means that there is a Turing machine solving the decision problem, but that there are some non-deterministic transitions in this machine.

- If a problem P_1 is NP-Complete, and we want to use it to prove that P_2 is also NP-Complete, in which direction is the reduction done (from an instance of P_1 to an instance of P_2 , or the other way around?)

It is necessary to propose a transformation of an instance of P_1 into an instance of P_2 , executable in polynomial time, such that the 'yes/no' answer is maintained between the two instances.

- What is the initial NP-Complete decision problem, from which all other NP-Complete problems are reduced in polynomial time?

The SAT problem.

- If an optimization problem admits an exponential number of solutions, is the associated decision problem NP-Complete?

No. Using a deterministic algorithm, we may be able to find the optimal solution without having to list them all (the Eulerian cycle problem of the previous prosit, for example, or the shortest path problem).

- Consider the graph n -colouring problem:

Given a graph G , can its vertices be coloured with only n colours, so that no neighbouring vertices have the same colour?

Is this problem in NP?

Yes. We only have to pass through all the vertices, check if the neighbourhood constraint is verified, and count the number of different colours. We have a complexity in $n \times \Delta(G) \leq n^2$ (reminder: $\Delta(G)$ is the maximum degree of graph G).

- Is the NP-Completeness proof of the classic VRP enough for the prosit issue?

In theory, no. The VRP considers a complete graph, while we have an incomplete graph. In order to have a problem in which an optimal solution can be built, distances corresponding to the shortest path between the vertices in the original graph must be added to these edges. However, by doing this, we are building a graph that has a very specific structure (it complies with the triangle inequality, see the definition of the metric VRP). This structure could be exploited to find an optimal solution in polynomial time, where the general case is NP-Complete.

In this case, the problem remains NP-Complete in this form, and the proof is exactly the same. But it is essential to understand that, in general, a very specific case of problem can be polynomial while it is NP-Complete in its general form.

- Does the NP-Completeness proof found prove that the problem is unsolvable?

No, it proves that solving the problem in a reasonable time is unlikely, unless you are a researcher with top level scientific skills).

Moreover, this proves that we cannot reasonably expect to maintain both the optimality and the polynomial complexity constraints at the same time. One of them can be waived (this is the topic of the next two prosits)

- If one day it is proved that $P=NP$, does this mean that any NP-Complete problem will be easily solved?

This proves that there is an algorithm of polynomial complexity that can solve all NP-Complete problems.

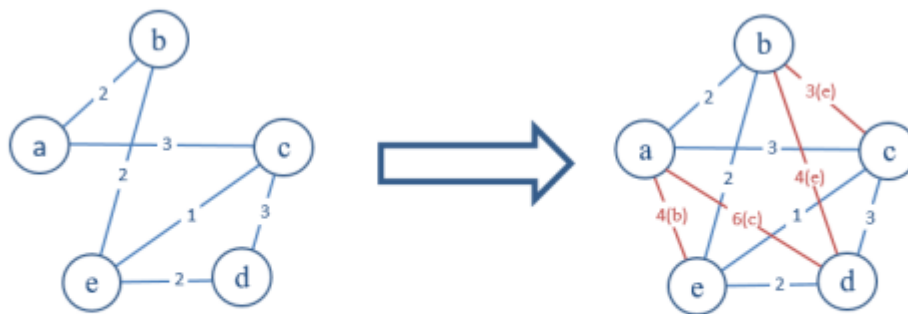
Advanced Algorithms Preparation Project 1 - PBL Loop 2 - PBL Loop 3 - PBL Loop 4 - PBL Loop

Below are some generic concepts that students should have understood by the end of the prosit. Answers may vary depending on the constraints chosen by the groups within the project scope. It would also be interesting to make students compare the answers in prosit outcome and feedback.

- **- Modelling the problem**

As long as we ignore cities without delivery, we are back to the metric version of the TSP:

- We model cities by vertices
- We model the paths between two cities by edges, weighted with the time required to complete the journey, connecting the corresponding vertices
- We add, between two unconnected vertices u and v , an edge e whose weight value is the shortest path (in terms of time) between u and v .



Transformation du problème concret de tournées d'entretien en un problème de TSP métrique (les chemins correspondant aux arêtes ajoutées sont précisés sur les distances de ces arêtes)

We thus obtain a complete graph complying with the triangle inequality, and whose length remains linear compared to the initial graph (the graph that exactly matches the road network). Calculating a Hamiltonian cycle of minimum length on this graph is basically authorising the same vertex to be passed through several times, which does not affect the optimality of the circuit in terms of length.

The formal definitions of the optimization problem and the associated optimization problem are provided above.

- **Determining the problem complexity**

The decision problem is NP-Complete. The proof is included in the definition of the problem, provided above.

- **Drawing a conclusion on the problem complexity**

Given that the Travelling Salesman Problem is NP-Complete, unless $P=NP$, it is impossible to find:

- An optimal solution
- A solution within a reasonable time, i.e. polynomial in the worst case

A polynomial complexity algorithm will necessarily be a heuristic, i.e. it will propose a solution that will be suboptimal for some instances. Given the context (computation performed in advance, on a potentially powerful computing machine), even a cubic complexity seems acceptable.

Advanced Algorithms Preparation Project 1 - PBL Loop 2 - PBL Loop 3 - PBL Loop 4 - PBL Loop

Note: the solution shown here concerns undirected graphs. For directed graphs, the principle is virtually the same.

WORKSHOP

TOPIC

✓ Instructions

QUIZ

Important: It is highly recommended that the exercise series associated with this loop is completed before tackling the workshop.

Workshop statement [doc] 

EDUCATIONAL GUIDE

✓ Standard corrections

Workshop version tuteur [doc] 


EXERCISE SERIES

TOPIC

✓ Resources for students

✓ Exercise series

Important: It is highly recommended that this exercise series is completed before tackling the workshop on this loop.

Statement and correction of the exercise series [pdf] 

EDUCATIONAL GUIDE

