La Recherche Opérationnelle ou l'art de bien optimiser: un panorama

Vincent T'kindt

Université de Tours
LIFAT(EA 6300), équipe ROOT (ERL CNRS 7002)
tkindt@univ-tours.fr

May 17, 2019

What is Operations Research?

"OR deals with the development of advanced analytical methods to solve decision or optimization problems",

What is Operations Research?

"*OR deals with the development of advanced analytical methods to solve decision or optimization problems*",

OR $\sim$ Combinatorial Optimization $\sim$ Discrete Optimization $\sim$ Continuous Optimization $\sim$ Mathematical Programming $\sim$ Constraint Programming $\sim$ ...

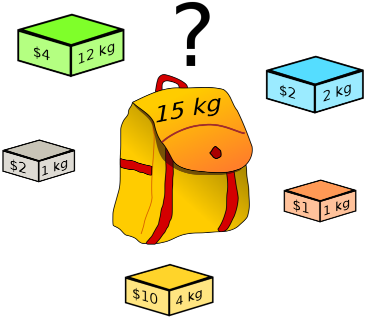
What is Operations Research?

"*OR deals with the development of advanced analytical methods to solve decision or optimization problems*",

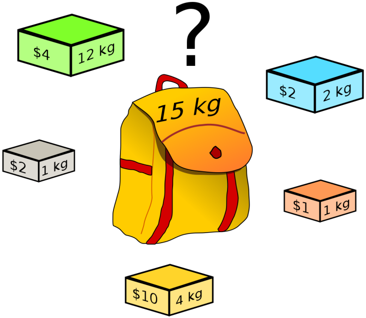
OR $\sim$ Combinatorial Optimization $\sim$ Discrete Optimization $\sim$ Continuous Optimization $\sim$ Mathematical Programming $\sim$ Constraint Programming $\sim$ ...

- **Main stream**: make use of mathematics and computer science to build appropriate **models** and **algorithms**.

Starting from an example



Starting from an example



0/1 KNAPSACK

Input: A finite set U, a size $s(u) \in \mathbb{N}$ and value $v(u) \in \mathbb{N}$ for each $u \in U$. A maximum size $B \in \mathbb{N}$.

Goal: Find a subset $U' \subseteq U$ such that

$$\sum_{u \in U'} s(u) \leq B \text{ and } \sum_{u \in U'} v(u) \text{ is maximum.}$$

Starting from an example

Maximize $\sum_{u \in U'} v(u)$
subject to
$\sum_{u \in U'} s(u) \leq B$
$U' \subseteq U$

Starting from an example

Maximize $\sum_{u \in U'} v(u)$
subject to
$\sum_{u \in U'} s(u) \leq B$
$U' \subseteq U$

Minimize $f(x)$
subject to
$x \in \mathcal{S}$

Starting from an example

$$\begin{array}{l} \text{Maximize } \sum_{u \in U'} v(u) \\ \quad \text{subject to} \\ \qquad \sum_{u \in U'} s(u) \leq B \\ \qquad U' \subseteq U \end{array}$$

$$\begin{array}{c} \text{Minimize } f(x) \\ \text{subject to} \\ x \in \mathcal{S} \end{array}$$

- How to solve this kind of problem?
  Exact/optimal algorithms $\Rightarrow$ optimal solutions.
  Heuristic algorithms $\Rightarrow$ "good" solutions.

Starting from an example

$$\begin{array}{l} \text{Maximize } \sum_{u \in U'} v(u) \\ \quad \text{subject to} \\ \qquad \sum_{u \in U'} s(u) \leq B \\ \qquad U' \subseteq U \end{array}$$

$$\begin{array}{c} \text{Minimize } f(x) \\ \text{subject to} \\ x \in \mathcal{S} \end{array}$$

- How to solve this kind of problem?
  Exact/optimal algorithms $\Rightarrow$ optimal solutions.
  Heuristic algorithms $\Rightarrow$ "good" solutions.
- Pt1: What is its complexity?

Starting from an example

$$\begin{array}{l} \text{Maximize } \sum_{u \in U'} v(u) \\ \text{subject to} \\ \quad \sum_{u \in U'} s(u) \leq B \\ \quad U' \subseteq U \end{array}$$

$$\begin{array}{c} \text{Minimize } f(x) \\ \text{subject to} \\ x \in \mathcal{S} \end{array}$$

- How to solve this kind of problem?
  Exact/optimal algorithms ⇒ optimal solutions.
  Heuristic algorithms ⇒ "good" solutions.
- Pt1: What is its complexity?
- Pt2: What is a good model?

Starting from an example

$$\begin{array}{l} \text{Maximize } \sum_{u \in U'} v(u) \\ \quad \text{subject to} \\ \qquad \sum_{u \in U'} s(u) \leq B \\ \qquad U' \subseteq U \end{array}$$

$$\begin{array}{c} \text{Minimize } f(x) \\ \text{subject to} \\ x \in \mathcal{S} \end{array}$$

- How to solve this kind of problem?
  Exact/optimal algorithms $\Rightarrow$ optimal solutions.
  Heuristic algorithms $\Rightarrow$ "good" solutions.
- Pt1: What is its complexity?
- Pt2: What is a good model?
- Pt3: Which solution algorithm?

What is Complexity Theory?

- Complexity theory provides tools to qualify the time complexity to make a computer solving a problem ([1]),

[1] Garey, S.M., Johnson, D.S. (1978). *Computers and Intractability*, W.H. Freeman and Company.

What is Complexity Theory?

- Complexity theory provides tools to qualify the time complexity to make a computer solving a problem ([1]),

[1] Garey, S.M., Johnson, D.S. (1978). *Computers and Intractability*, W.H. Freeman and Company.

What is Complexity Theory?

- Complexity theory provides tools to qualify the time complexity to make a computer solving a problem ([1]),
- **0/1 KNAPSACK**: An *instance* I is a tuple (U, s, v, B),

[1] Garey, S.M., Johnson, D.S. (1978). *Computers and Intractability*, W.H. Freeman and Company.

What is Complexity Theory?

- Complexity theory provides tools to qualify the time complexity to make a computer solving a problem ([1]),
- **0/1 KNAPSACK**: An *instance* I is a tuple (U, s, v, B),
- The *size* n of I is the number of elements, *i.e.* $n = |U|$,

[1] Garey, S.M., Johnson, D.S. (1978). *Computers and Intractability*, W.H. Freeman and Company.

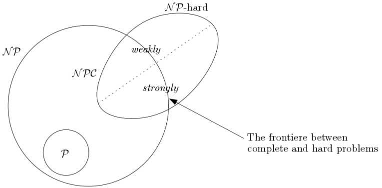
What is Complexity Theory?

- Complexity theory provides tools to qualify the time complexity to make a computer solving a problem ([1]),
- **0/1 KNAPSACK**: An *instance* I is a tuple (U, s, v, B),
- The *size* n of I is the number of elements, *i.e.* $n = |U|$,
- What is the smallest time complexity (in the worst case) a computer can achieve to solve the **0/1 KNAPSACK**?

[1] Garey, S.M., Johnson, D.S. (1978). *Computers and Intractability*, W.H. Freeman and Company.

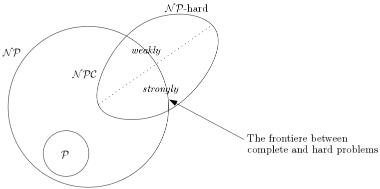
Complexity classes

- The complexity Zoo (`complexityzoo.uwaterloo.ca/Complexity_Zoo`),



Complexity classes

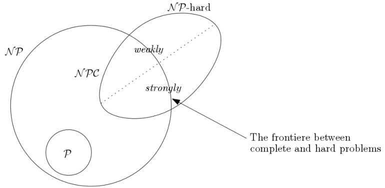
- The complexity Zoo (`complexityzoo.uwaterloo.ca/Complexity_Zoo`),



- Class $\mathcal{P}$: contains problems solvable in polynomial time of the instance size n (easy problems),

Complexity classes

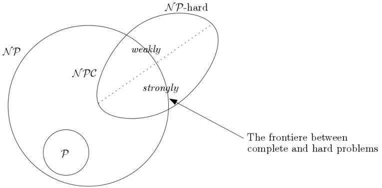
- The complexity Zoo (`complexityzoo.uwaterloo.ca/Complexity_Zoo`),



- Class $\mathcal{P}$: contains problems solvable in polynomial time of the instance size n (easy problems),
- Class $\mathcal{NPC}$: contains problems **not** solvable in polynomial time of the instance size n (hard problems),

Complexity classes

- The complexity Zoo (`complexityzoo.uwaterloo.ca/Complexity_Zoo`),



- Class $\mathcal{P}$: contains problems solvable in polynomial time of the instance size n (easy problems),
- Class $\mathcal{NPC}$: contains problems **not** solvable in polynomial time of the instance size n (hard problems),
- $\mathcal{P}$ vs $\mathcal{NP}$? Assumption: $\mathcal{P} \neq \mathcal{NP}$,

  One of the millennium problems of the Clay Mathematics Institute ($1 million reward)

Consequences

- What can we do under the previous hypothesis?

Consequences

- What can we do under the previous hypothesis?
- For problems shown to be in $\mathcal{P}$: find an optimal algorithm running in polynomial time of n,

Consequences

- What can we do under the previous hypothesis?
- For problems shown to be in $\mathcal{P}$: find an optimal algorithm running in polynomial time of n,
- For problems shown to be in $\mathcal{NPC}$ ($\mathcal{NP}$-hard problems):

Consequences

- What can we do under the previous hypothesis?
- For problems shown to be in $\mathcal{P}$: find an optimal algorithm running in polynomial time of n,
- For problems shown to be in $\mathcal{NPC}$ ($\mathcal{NP}$-hard problems):
    1. Option 1: Find an optimal algorithm with the "lowest possible" time complexity (though exponential in n)... or at least, fast enough in practice.

Consequences

- What can we do under the previous hypothesis?
- For problems shown to be in $\mathcal{P}$: find an optimal algorithm running in polynomial time of n,
- For problems shown to be in $\mathcal{NPC}$ ($\mathcal{NP}$-hard problems):
  1. Option 1: Find an optimal algorithm with the "lowest possible" time complexity (though exponential in n)... or at least, fast enough in practice.
  2. Option 2: Find a heuristic algorithm running in polynomial time (so, no guarantee to have the optimal solution).

A scheduling problem

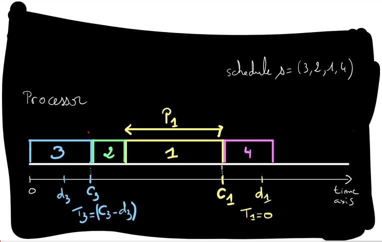
- Getting a model leading to an effective solution is fundamental!

A scheduling problem

- Getting a model leading to an effective solution is fundamental!
- Consider the following scheduling problem,

Single machine total tardiness (SMTT)

Let be n tasks to perform on a single processor. Each task j is defined by a known processing time p_j and a due date d_j.
For a given schedule s, each task j is given a completion time C_j and a tardiness $T_j = \max(0; C_j - d_j)$.
Find a schedule s with minimum $\sum_j T_j$ value.



A scheduling problem

- This problem is $\mathcal{NP}$-hard in the ordinary sense,

A scheduling problem

- This problem is $\mathcal{NP}$-hard in the ordinary sense,
- Mathematical Programming (MP) to model the problem,

A scheduling problem

- This problem is $\mathcal{NP}$-hard in the ordinary sense,
- Mathematical Programming (MP) to model the problem,
- A position based IP formulation (IP_1),

$$
\begin{array}{lr}
\text{Minimize } \sum_{k=1}^{n} T_{[k]} & \\
\quad \text{s.t.} & \\
\qquad \sum_{k=1} x_{j,k} = 1 & \forall j = 1, ..., n \\
\qquad \sum_{j=1} x_{j,k} = 1 & \forall k = 1, ..., n \\
\qquad T_{[k]} \geq \sum_{\ell=1}^{k} \sum_{j=1}^{n} x_{j,\ell} p_j - \sum_{j=1}^{n} x_{j,k} d_j & \forall k = 1, ..., n \\
\qquad T_{[k]} \geq 0 & \forall k = 1, ..., n \\
\qquad x_{j,k} \in \{0; 1\} & \forall j, k
\end{array}
$$

A scheduling problem

- This problem is $\mathcal{NP}$-hard in the ordinary sense,
- Mathematical Programming (MP) to model the problem,
- A position based IP formulation (IP_1),

$$
\begin{aligned}
&\text{Minimize } \sum_{k=1}^{n} T_{[k]} \\
&\quad \text{s.t.} \\
&\qquad \sum_{k=1} x_{j,k} = 1 && \forall j = 1, ..., n \\
&\qquad \sum_{j=1} x_{j,k} = 1 && \forall k = 1, ..., n \\
&\qquad T_{[k]} \geq \sum_{\ell=1}^{k} \sum_{j=1}^{n} x_{j,\ell} p_j - \sum_{j=1}^{n} x_{j,k} d_j && \forall k = 1, ..., n \\
&\qquad T_{[k]} \geq 0 && \forall k = 1, ..., n \\
&\qquad x_{j,k} \in \{0; 1\} && \forall j, k
\end{aligned}
$$

- For a given instance, provide (IP_1) to a commercial solver (e.g. CPLEX, Gurobi, XPress) and press the *Solve* button!

A scheduling problem

- This problem is $\mathcal{NP}$-hard in the ordinary sense,
- Mathematical Programming (MP) to model the problem,
- A position based IP formulation (IP_1),

$$
\begin{aligned}
&\text{Minimize } \sum_{k=1}^{n} T_{[k]} \\
&\quad \text{s.t.} \\
&\qquad \sum_{k=1} x_{j,k} = 1 && \forall j = 1, ..., n \\
&\qquad \sum_{j=1} x_{j,k} = 1 && \forall k = 1, ..., n \\
&\qquad T_{[k]} \geq \sum_{\ell=1}^{k} \sum_{j=1}^{n} x_{j,\ell} p_j - \sum_{j=1}^{n} x_{j,k} d_j && \forall k = 1, ..., n \\
&\qquad T_{[k]} \geq 0 && \forall k = 1, ..., n \\
&\qquad x_{j,k} \in \{0; 1\} && \forall j, k
\end{aligned}
$$

- For a given instance, provide (IP_1) to a commercial solver (e.g. CPLEX, Gurobi, XPress) and press the *Solve* button!
- You should be able to solve instances up to about 50 jobs in size.

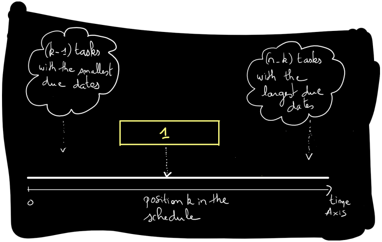
A scheduling problem

- Model the problem differently by making use of Lawler's decomposition [2],

[2] Lawler, E.L (1977). *A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness*, Annals of Discrete Mathematics, 1:331-342.

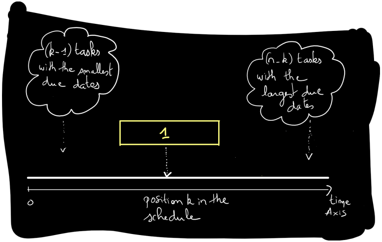
A scheduling problem

- Model the problem differently by making use of Lawler's decomposition [2],
- Illustration on an example,



A scheduling problem

- Model the problem differently by making use of Lawler's decomposition [2],
- Illustration on an example,



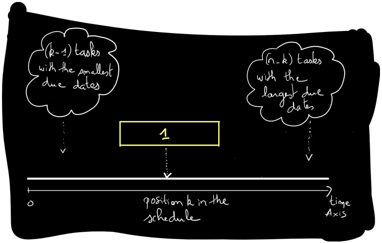
- Dynamic Programming (DP):

$$T[S,0] = \min_{s\leq \ell \leq e}(T[B_\ell,0] + T[A_\ell, \sum_{j\in B_\ell \cup \{j^*\}} p_j] + \max(0; \sum_{j\in B_\ell \cup \{j^*\}} p_j - d_{j^*}))$$

with j^* the longest task in S, B_ℓ (resp. A_ℓ) tasks before j^* (resp. after) when sequenced in position ℓ.

A scheduling problem

- Model the problem differently by making use of Lawler's decomposition [2],
- Illustration on an example,



- Dynamic Programming (DP):

$$T[S,0] = \min_{s \leq \ell \leq e}\left(T[B_\ell,0] + T[A_\ell, \textstyle\sum_{j \in B_\ell \cup \{j^*\}} p_j] + \max(0; \textstyle\sum_{j \in B_\ell \cup \{j^*\}} p_j - d_{j^*})\right)$$

with j^* the longest task in S, B_ℓ (resp. A_ℓ) tasks before j^* (resp. after) when sequenced in position ℓ.

- You should be able to solve instances up to 100 tasks in size.

Intermediate conclusions

Pit stop

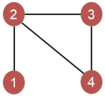
- MP and DP are approaches usable for building models,
- MP and DP can be used also to solve your problem,
- The effectiveness of MP solvers is continuously improved: good challengers for the exact solution of decision/optimization problems.

The GED problem

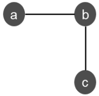
- Another example: The Graph Edit Distance problem (GED),

The GED problem

- Another example: The Graph Edit Distance problem (GED),



G



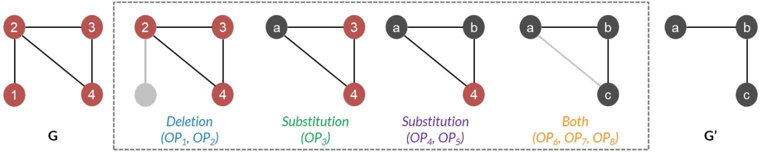
G’

The GED problem

Let $G = (V, E, \mu, \xi)$ and $G' = (V', E', \mu', \xi')$ be two undirected attributed graphs, with μ (resp. μ') is the set of labels attached to vertices in V (resp. V'), and ξ (resp. ξ') is the set of labels attached to edges in E (resp E').
Let λ be and *edit path*: a minimal set of operations (deletion, insertion, substitution) to transform G into G'.
Find λ^* with minimal cost $d(\lambda) = \sum_{o \in \lambda} c(o)$.

The GED problem

- Another example: The Graph Edit Distance problem (GED),



The GED problem

Let $G = (V, E, \mu, \xi)$ and $G' = (V', E', \mu', \xi')$ be two undirected attributed graphs, with μ (resp. μ') is the set of labels attached to vertices in V (resp. V'), and ξ (resp. ξ') is the set of labels attached to edges in E (resp E').

Let λ be and *edit path*: a minimal set of operations (deletion, insertion, substitution) to transform G into G'.

Find λ^* with minimal cost $d(\lambda) = \sum_{o \in \lambda} c(o)$.

The GED problem

- The GED problem is $\mathcal{NP}$-hard,

[3] Lerouge, J. Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., Héroux, P., Adam, S. (2017). *New binary linear programming formulation to compute the graph edit distance*, Pattern Recognition, 72:254-265.

The GED problem

- The GED problem is $\mathcal{NP}$-hard,
- Consider a first IP formulation of the problem (IP1) [3],

[3] Lerouge, J. Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., Héroux, P., Adam, S. (2017). *New binary linear programming formulation to compute the graph edit distance*, Pattern Recognition, 72:254-265.

The GED problem

- The GED problem is $\mathcal{NP}$-hard,
- Consider a first IP formulation of the problem (IP1) [3],
  - Variables:

$$x_{i,k} = \begin{cases} 1 & \text{if } i \in V \text{ is matched with } k \in V' \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ij,k\ell} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ is matched with } (k,\ell) \in E' \\ 0 & \text{otherwise} \end{cases}$$

$|V| \times |V'|$ variables $x_{i,k}$ and $|E| \times |E'|$ variables $y_{ij,k\ell}$.

[3] Lerouge, J. Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., Héroux, P., Adam, S. (2017). *New binary linear programming formulation to compute the graph edit distance*, Pattern Recognition, 72:254-265.

The GED problem

- Consider a first IP formulation of the problem (IP1) [3],

[3] Lerouge, J. Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., Héroux, P., Adam, S. (2017). *New binary linear programming formulation to compute the graph edit distance*, Pattern Recognition, 72:254-265.

The GED problem

- Consider a first IP formulation of the problem (IP1) [3],
  - Constraints:

$$\sum_{k\in V'} x_{i,k} \leq 1 \qquad \forall i \in V \quad \text{(A)}$$
$$\sum_{i\in V} x_{i,k} \leq 1 \qquad \forall k \in V' \quad \text{(B)}$$

$$\sum_{(k,\ell)\in E'} y_{ij,k\ell} \leq x_{i,k} + x_{j,k} \qquad \forall k \in V', \forall (i,j) \in E \quad \text{(C)}$$

[3] Lerouge, J. Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., Héroux, P., Adam, S. (2017). *New binary linear programming formulation to compute the graph edit distance*, Pattern Recognition, 72:254-265.

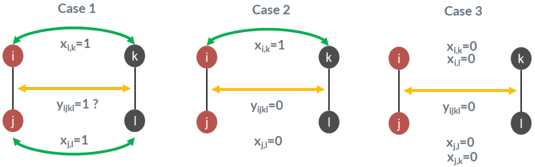
The GED problem

- Consider a first IP formulation of the problem (IP1) [3],
  - Constraints:

$$\sum_{k\in V'} x_{i,k} \leq 1 \qquad \forall i \in V \quad (A)$$
$$\sum_{i\in V} x_{i,k} \leq 1 \qquad \forall k \in V' \quad (B)$$

$$\sum_{(k,\ell)\in E'} y_{ij,k\ell} \leq x_{i,k} + x_{j,k} \qquad \forall k \in V', \forall (i,j) \in E \quad (C)$$

Ex: $y_{ij,k\ell} \leq x_{i,k} + x_{j,k}$ and $y_{ij,k\ell} \leq x_{i,\ell} + x_{j,\ell}$



[3] Lerouge, J. Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., Héroux, P., Adam, S. (2017). *New binary linear programming formulation to compute the graph edit distance*, Pattern Recognition, 72:254-265.

The GED problem

- Consider a first IP formulation of the problem (IP1) [3],
  - <u>Constraints:</u>

$$\sum_{k\in V'} x_{i,k} \leq 1 \qquad \forall i \in V \qquad \text{(A)}$$
$$\sum_{i\in V} x_{i,k} \leq 1 \qquad \forall k \in V' \qquad \text{(B)}$$

$$\sum_{(k,\ell)\in E'} y_{ij,k\ell} \leq x_{i,k} + x_{j,k} \qquad \forall k \in V', \forall (i,j) \in E \qquad \text{(C)}$$

$|V| + |V'| + |V'| \times |E|$ constraints.

[3] Lerouge, J. Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., Héroux, P., Adam, S. (2017). *New binary linear programming formulation to compute the graph edit distance*, Pattern Recognition, 72:254-265.

The GED problem

- Consider a first IP formulation of the problem (IP1) [3],
  - Objective function:
    Minimize
    $\sum_{i \in V} \sum_{k \in V'} C_v(i,k) x_{i,k} + \sum_{(i,j) \in E} \sum_{(k),\ell \in E'} C_e(ij, k\ell) y_{ij,k\ell} + CSTE$

[3] Lerouge, J. Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., Héroux, P., Adam, S. (2017). *New binary linear programming formulation to compute the graph edit distance*, Pattern Recognition, 72:254-265.

The GED problem

- Consider a second IP formulation of the problem (IP2) [4],

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

The GED problem

- Consider a second IP formulation of the problem (IP2) [4],
  - Variables:
    Same $x_{i,k}$ variables.

$$y_{ij,k\ell} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ is matched with } (k,\ell) \in \tilde{E}' \\ 0 & \text{otherwise} \end{cases}$$

with $\tilde{E}' = E' \cup \{(\ell, k)/(k, \ell) \in E'\}$
$|E| \times |E'|$ variables $x_{i,k}$ and $2 \times |V| \times |V'|$ variables $y_{ij,k\ell}$.

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

The GED problem

- Consider a second IP formulation of the problem (IP2) [4],

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

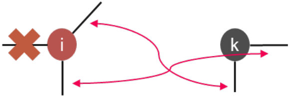
The GED problem

- Consider a second IP formulation of the problem (IP2) [4],
  - Constraints:
    Constraints (A) and (B)
    We change constraints (C)

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

The GED problem

- Consider a second IP formulation of the problem (IP2) [4],
  - Constraints:
    Constraints (A) and (B)
    We change constraints (C)

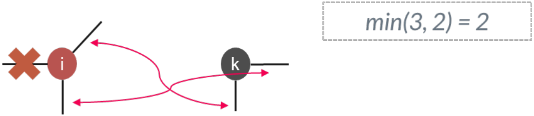


$min(3, 2) = 2$

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

The GED problem

- Consider a second IP formulation of the problem (IP2) [4],
  - <u>Constraints:</u>
    Constraints (A) and (B)
    We change constraints (C)

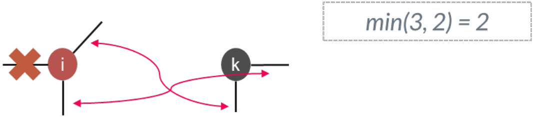


$$\sum_{(i,j)\in E} \sum_{(k,\ell)\in \tilde{E}'} y_{ij,k\ell} \leq d_{i,k} x_{i,k} \qquad \forall i \in V, \forall k \in V' \qquad \text{(D)}$$

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

The GED problem

- Consider a second IP formulation of the problem (IP2) [4],
  - Constraints:
    Constraints (A) and (B)
    We change constraints (C)



$$\sum_{(i,j)\in E} \sum_{(k,\ell)\in \tilde{E}'} y_{ij,k\ell} \leq d_{i,k} x_{i,k} \qquad \forall i \in V, \forall k \in V' \quad \text{(D)}$$

$|V| + |V'| + |V| \times |V'|$ constraints.

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

The GED problem

- (IP2) has more variables but less constraints than (IP1),

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

The GED problem

- (IP2) has more variables but less constraints than (IP1),
- What's the impact on a computational side?

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

The GED problem

- (IP2) has more variables but less constraints than (IP1),
- What's the impact on a computational side?
- Tests done on 660 instances from CMUHOUSE database [4] (30 vertices per graph),

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

The GED problem

- (IP2) has more variables but less constraints than (IP1),
- What's the impact on a computational side?
- Tests done on 660 instances from CMUHOUSE database [4] (30 vertices per graph),

model (IP1)		model (IP2)	
t_{avg} (s)	$\#Opt$	t_{avg} (s)	$\#Opt$
395.33	25	20.26	25

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

The GED problem

- (IP2) has more variables but less constraints than (IP1),
- What's the impact on a computational side?
- Tests done on 660 instances from CMUHOUSE database [4] (30 vertices per graph),

model (IP1)			model (IP2)		
t_{avg} (s)	$\#Opt$	d_{avg} (%)	t_{avg} (s)	$\#Opt$	d_{avg} (%)
880.74	25	604.11	497.07	365	0.70

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

Final conclusions

Conclusion

Thinking about the model is as less as important as thinking about solution algorithms.
Designing models occur when dealing with a problem... but also when designing optimization algorithms.

Introduction

- Depending on the complexity of the problem: Exact or Heuristic algorithms,

Introduction

- Depending on the complexity of the problem: Exact or Heuristic algorithms,
- The toolbox of OR (non exhaustive),

Exact algorithms		Heuristic algorithms	
Family	Name	Family	Name
Branching algorithms	Branch-and-Bound	Constructive algorithms	Priority based
	Branch-and-cut		Greedy
	Branch-and-price		Ant CO
	Branch-and-cut-and-price	Branching algorithms	Beam Search
Mathematical Programming	LP		Recovering BS
	MILP		Branch-and-Greed
	QP		Limited Discrepancy Se
	SDP	Neighborhood based algorithms	Simulated Annealing
Dynamic Programming	Forward DP		Tabu
	Backward DP		Multistart
	DP across the subsets		VNS
Constraint Programming			GRASP
Dedicated Approaches			Genetic Algorithms
			Bees Algorithms
		Matheuristics	VPLS
			Local Branching

What's next?

Golden rules

1. Never design an exponential-time exact or heuristic algorithm for a problem in class $\mathcal{P}$,

What's next?

Golden rules

1. Never design an exponential-time exact or heuristic algorithm for a problem in class $\mathcal{P}$,
2. If your problem is in class $\mathcal{P}$, find the right polynomial-time exact algorithm (dedicated),

- We will see some of the most interesting approaches (to my opinion),

What's next?

Golden rules

1. Never design an exponential-time exact or heuristic algorithm for a problem in class $\mathcal{P}$,
2. If your problem is in class $\mathcal{P}$, find the right polynomial-time exact algorithm (dedicated),
3. If your problem is in class $\mathcal{NPC}$, don't search for a polynomial-time exact algorithm: optimality $\Rightarrow$ exponentiality. Do heuristics?

- We will see some of the most interesting approaches (to my opinion),

Exact algorithms for $\mathcal{NP}$-hard problems

- Commercial solvers (CPLEX, Gurobi, XPress) of MILP are now very competitive,

Exact algorithms for $\mathcal{NP}$-hard problems

- Commercial solvers (CPLEX, Gurobi, XPress) of MILP are now very competitive,
- First design a good MILP model of your problem,

Exact algorithms for $\mathcal{NP}$-hard problems

- Commercial solvers (CPLEX, Gurobi, XPress) of MILP are now very competitive,
- First design a good MILP model of your problem,
- Solve it thanks to a commercial solver,

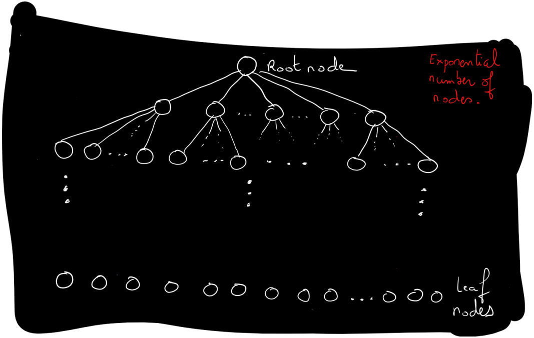
Exact algorithms for $\mathcal{NP}$-hard problems

- Commercial solvers (CPLEX, Gurobi, XPress) of MILP are now very competitive,
- First design a good MILP model of your problem,
- Solve it thanks to a commercial solver,
- Try to design a more effective exact algorithm,

  NB: "more effective" means capable of solving to optimality instances of largest size.

Exact solution of the SMTT problem

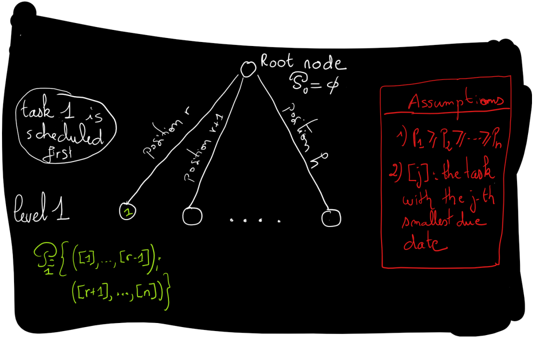
- Based on Lawler's decomposition, we have designed an exact branching algorithm [5],



[5] Shang, L., T'kindt, V., Della Croce, F. (2018). *The Memorization Paradigm: Branch and Memorize algorithms for the efficient solution of sequencing problems.* https://hal.archives-ouvertes.fr/hal-01599835.

Exact solution of the SMTT problem

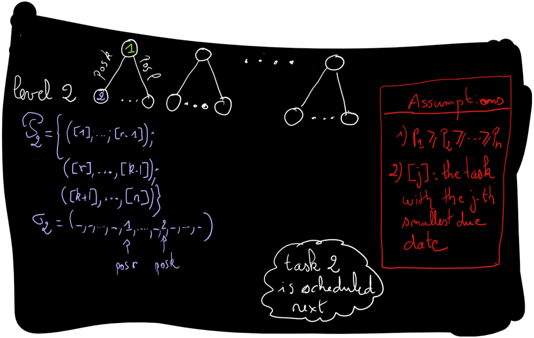
- Based on Lawler's decomposition, we have designed an exact branching algorithm [5],



[5] Shang, L., T'kindt, V., Della Croce, F. (2018). *The Memorization Paradigm: Branch and Memorize algorithms for the efficient solution of sequencing problems.* https://hal.archives-ouvertes.fr/hal-01599835.

Exact solution of the SMTT problem

- Based on Lawler's decomposition, we have designed an exact branching algorithm [5],



[5] Shang, L., T'kindt, V., Della Croce, F. (2018). *The Memorization Paradigm: Branch and Memorize algorithms for the efficient solution of sequencing problems.* https://hal.archives-ouvertes.fr/hal-01599835.

Exact solution of the SMTT problem

- In Branch-and-Bound algorithms, we also add a *bounding mechanism*:

Exact solution of the SMTT problem

- In Branch-and-Bound algorithms, we also add a *bounding mechanism*:
  1. First, compute a global upper bound UB,

Exact solution of the SMTT problem

- In Branch-and-Bound algorithms, we also add a *bounding mechanism*:

  1. First, compute a global upper bound UB,
  2. At each node s, compute a lower bound $LB(s)$ to the best solution that can be built from s,

Exact solution of the SMTT problem

- In Branch-and-Bound algorithms, we also add a *bounding mechanism*:

  1. First, compute a global upper bound UB,
  2. At each node s, compute a lower bound $LB(s)$ to the best solution that can be built from s,
  3. If $(LB(s) > UB)$ then prune node s.

Exact solution of the SMTT problem

- In Branch-and-Bound algorithms, we also add a *bounding mechanism*:
  1. First, compute a global upper bound UB,
  2. At each node s, compute a lower bound $LB(s)$ to the best solution that can be built from s,
  3. If $(LB(s) > UB)$ then prune node s.
- For the SMTT problem, the bouding mechanism was useless due to the presence of a *memorization mechanism*,

Exact solution of the SMTT problem

- In Branch-and-Bound algorithms, we also add a *bounding mechanism*:
  1. First, compute a global upper bound UB,
  2. At each node s, compute a lower bound $LB(s)$ to the best solution that can be built from s,
  3. If $(LB(s) > UB)$ then prune node s.
- For the SMTT problem, the bouding mechanism was useless due to the presence of a *memorization mechanism*,
- In Branch-and-X algorithms, we can also add *cuts*:

If $(C_1(r) > d_{[r+1]})$ then there is no optimal solution in which task 1 is scheduled in position r,

with $C_1(r)$ the completion time of task 1 in the EDD schedule when task 1 is moved to position r.

Exact solution of the SMTT problem

- For the SMTT problem, a bunch of cuts is used (all forbid positions),

Exact solution of the SMTT problem

- For the SMTT problem, a bunch of cuts is used (all forbid positions),
- Computational experiments show that we solve instances with up to 500 tasks,

  MIP: 50 tasks / DP: 100 tasks

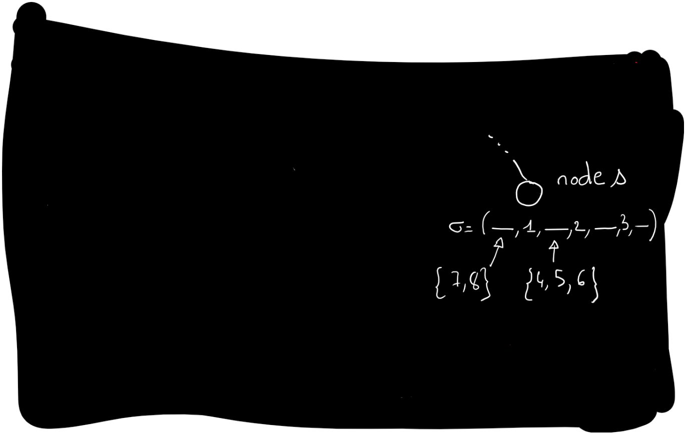
Exact solution of the SMTT problem

- For the SMTT problem, a bunch of cuts is used (all forbid positions),
- Computational experiments show that we solve instances with up to 500 tasks,

  MIP: 50 tasks / DP: 100 tasks
- We improve these results by adding a *memorization mechanism*: remember the exploration you have done so far, to avoid exploring useless subproblems in the future.

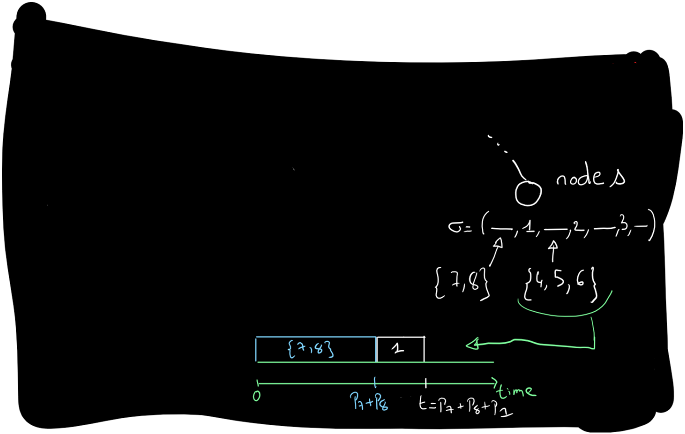
Exact solution of the SMTT problem

- The *memorization mechanism*,



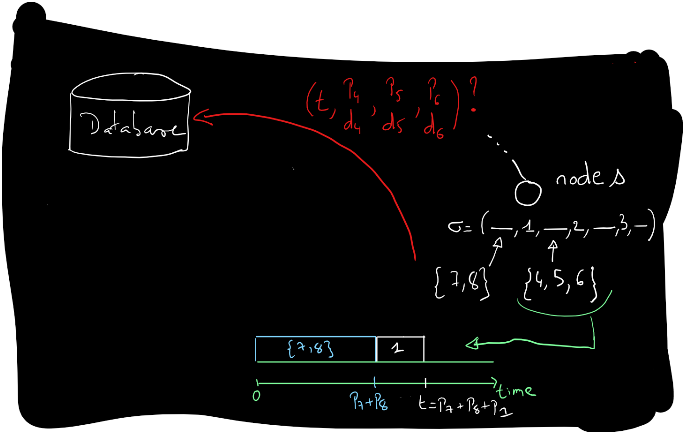
Exact solution of the SMTT problem

- The *memorization mechanism*,



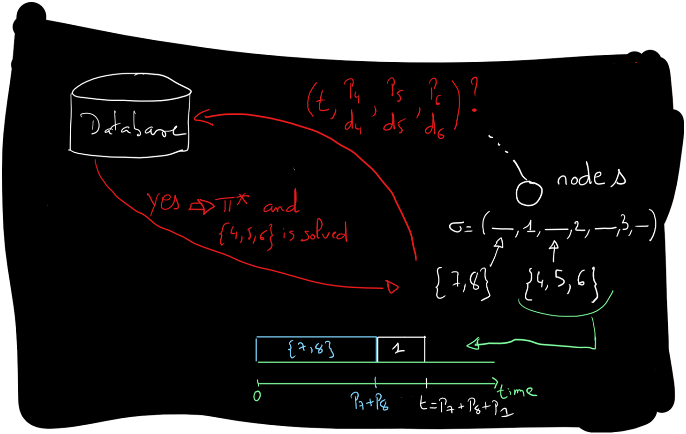
Exact solution of the SMTT problem

- The *memorization mechanism*,



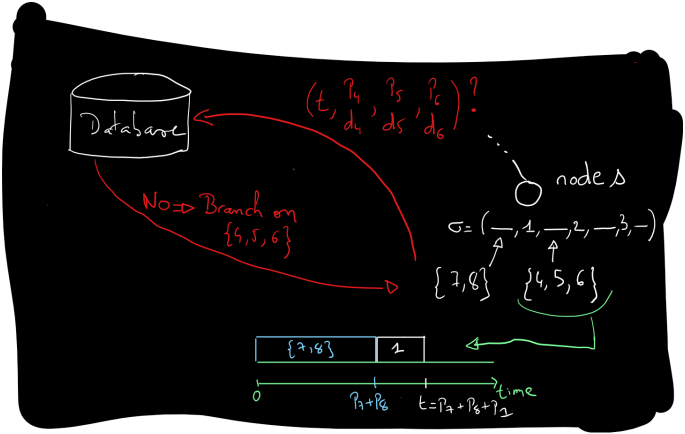
Exact solution of the SMTT problem

- The *memorization mechanism*,



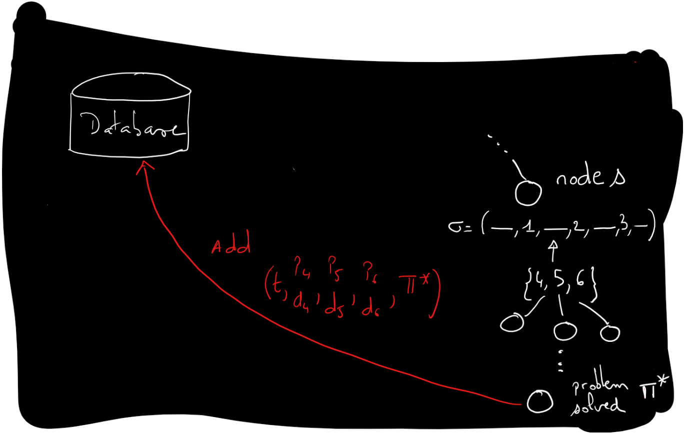
Exact solution of the SMTT problem

- The *memorization mechanism*,



Exact solution of the SMTT problem

- The *memorization mechanism*,



Exact solution of the SMTT problem

- Management of the database is a crucial point,

Exact solution of the SMTT problem

- Management of the database is a crucial point,
- Computational experiments show that we solve instances with up to 1200 tasks,

  MIP: 50 tasks / DP: 100 tasks / B&B: 500 tasks

Exact solution of the SMTT problem

- Management of the database is a crucial point,
- Computational experiments show that we solve instances with up to 1200 tasks,

  MIP: 50 tasks / DP: 100 tasks / B&B: 500 tasks

Pit stop

We have seen so far different exact approaches for $\mathcal{NP}$-hard optimization problems:

- Mathematical Programming (MILP),
- Dynamic Programming (DP),
- Branch-and-X algorithms.

From exact to heuristic approaches

- Finding exact algorithms is a very challenging issue,

From exact to heuristic approaches

- Finding exact algorithms is a very challenging issue,
- For $\mathcal{NP}$-hard problems, we are faced with combinatorics $\Rightarrow$ CPU times may become quickly non acceptable,

From exact to heuristic approaches

- Finding exact algorithms is a very challenging issue,
- For $\mathcal{NP}$-hard problems, we are faced with combinatorics $\Rightarrow$ CPU times may become quickly non acceptable,
- Heuristic approaches may become the only option,

From exact to heuristic approaches

- Finding exact algorithms is a very challenging issue,
- For $\mathcal{NP}$-hard problems, we are faced with combinatorics $\Rightarrow$ CPU times may become quickly non acceptable,
- Heuristic approaches may become the only option,
- A heuristic algorithm $=$ polynomial running time but no warranty of computing the optimal solution,

From exact to heuristic approaches

- Finding exact algorithms is a very challenging issue,
- For $\mathcal{NP}$-hard problems, we are faced with combinatorics $\Rightarrow$ CPU times may become quickly non acceptable,
- Heuristic approaches may become the only option,
- A heuristic algorithm $=$ polynomial running time but no warranty of computing the optimal solution,
- The challenge: (i) find heuristics as close as possible to the optimal solution, (ii) acceptable running time.

Heuristic solution of the GED problem

- Many heuristics have been designed for that problem (see e.g. [4]),

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

[6] Bougleux, S., Brun, L., Carletti, V., Foggia, P., Gauzere, B., Vento, M. (2017). *Graph edit distance as a quadratic assignment problem*, Pattern Recognition Letters, 87:38-46.

[7] Brun, L. (2017). *Graph edit distance: Basics and History*, Workshop on Graph-based Representations in Pattern Recognition (GbR 17), Capri (Italy).

Heuristic solution of the GED problem

- Many heuristics have been designed for that problem (see e.g. [4]),
- Two efficient ones:
  - IPFP ([6]): *neighborhood based algorithm* which improves an initial solution by a local search phase in continuous space (QAP),

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

[6] Bougleux, S., Brun, L., Carletti, V., Foggia, P., Gauzere, B., Vento, M. (2017). *Graph edit distance as a quadratic assignment problem*, Pattern Recognition Letters, 87:38-46.

[7] Brun, L. (2017). *Graph edit distance: Basics and History*, Workshop on Graph-based Representations in Pattern Recognition (GbR 17), Capri (Italy).

Heuristic solution of the GED problem

- Many heuristics have been designed for that problem (see e.g. [4]),
- Two efficient ones:
  - IPFP ([6]): *neighborhood based algorithm* which improves an initial solution by a local search phase in continuous space (QAP),
  - GNCCP ([6]): *neighborhood based algorithm* intensively using IPFP on reformulations of the QAP.

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

[6] Bougleux, S., Brun, L., Carletti, V., Foggia, P., Gauzere, B., Vento, M. (2017). *Graph edit distance as a quadratic assignment problem*, Pattern Recognition Letters, 87:38-46.

[7] Brun, L. (2017). *Graph edit distance: Basics and History*, Workshop on Graph-based Representations in Pattern Recognition (GbR 17), Capri (Italy).

Heuristic solution of the GED problem

- Many heuristics have been designed for that problem (see e.g. [4]),
- Two efficient ones:
  - IPFP ([6]): *neighborhood based algorithm* which improves an initial solution by a local search phase in continuous space (QAP),
  - GNCCP ([6]): *neighborhood based algorithm* intensively using IPFP on reformulations of the QAP.
- Other heuristics exist (some based on branching approaches) but are less efficient than IPFP or GNCCP ([4, 6, 7]),

[4] Darwiche, M. (2018). *When Operations Research meets Structural Pattern Recognition: on the solution of Error-Tolerant Graph Matching Problems*, Ph.D. Thesis, University of Tours (France).

[6] Bougleux, S., Brun, L., Carletti, V., Foggia, P., Gauzere, B., Vento, M. (2017). *Graph edit distance as a quadratic assignment problem*, Pattern Recognition Letters, 87:38-46.

[7] Brun, L. (2017). *Graph edit distance: Basics and History*, Workshop on Graph-based Representations in Pattern Recognition (GbR 17), Capri (Italy).

Heuristic solution of the GED problem

- What's the problem with the previous heuristics?

Heuristic solution of the GED problem

- What's the problem with the previous heuristics?
- QAP may not be a good choice (there are linearities to exploit in the problem): we have good MILP formulations,

Heuristic solution of the GED problem

- What's the problem with the previous heuristics?
- QAP may not be a good choice (there are linearities to exploit in the problem): we have good MILP formulations,
- Using continuous relaxations to explore a discrete set of solutions is not always a good idea,

Heuristic solution of the GED problem

- What's the problem with the previous heuristics?
- QAP may not be a good choice (there are linearities to exploit in the problem): we have good MILP formulations,
- Using continuous relaxations to explore a discrete set of solutions is not always a good idea,
- Mathematical Programming strongly exploits the powerfulness of branching algorithms and polyhedral properties,

Heuristic solution of the GED problem

- What's the problem with the previous heuristics?
- QAP may not be a good choice (there are linearities to exploit in the problem): we have good MILP formulations,
- Using continuous relaxations to explore a discrete set of solutions is not always a good idea,
- Mathematical Programming strongly exploits the powerfulness of branching algorithms and polyhedral properties,
- We will see a *neighborhood based heuristic* based on Mathematical Programming: *Matheuristics*.

A Matheuristic for the GED problem

- We design a Local Branching heuristic (LocBra, [8,9])

[8] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). *A local branching heuristic for solving a graph edit distance problem*, Computers & Operations Research, 106:225-235.
[9] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). *Graph Edit Distance: Accuracy of Local Branching from an application point of view*, Pattern Recognition Letters, in press.

A Matheuristic for the GED problem

- We design a Local Branching heuristic (LocBra, [8,9])
- We make use of (IP2) formulation,

$$x_{i,k} = \begin{cases} 1 & \text{if } i \in V \text{ is matched with } k \in V' \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ij,k\ell} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ is matched with } (k,\ell) \in \tilde{E}' \\ 0 & \text{otherwise} \end{cases}$$

[8] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). *A local branching heuristic for solving a graph edit distance problem*, Computers & Operations Research, 106:225-235.

[9] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). *Graph Edit Distance: Accuracy of Local Branching from an application point of view*, Pattern Recognition Letters, in press.

A Matheuristic for the GED problem

- We design a Local Branching heuristic (LocBra, [8,9])
- We make use of (IP2) formulation,

$$x_{i,k} = \begin{cases} 1 & \text{if } i \in V \text{ is matched with } k \in V' \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ij,k\ell} = \begin{cases} 1 & \text{if } (i,j) \in E \text{ is matched with } (k,\ell) \in \tilde{E}' \\ 0 & \text{otherwise} \end{cases}$$

- We only work on the $x_{i,k}$ variables,

[8] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). *A local branching heuristic for solving a graph edit distance problem*, Computers & Operations Research, 106:225-235.
[9] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). *Graph Edit Distance: Accuracy of Local Branching from an application point of view*, Pattern Recognition Letters, in press.

A Matheuristic for the GED problem

- We start with an initial solution x^0 (*e.g.* solve (IP2) for $t = 180$s),

A Matheuristic for the GED problem

- We start with an initial solution x^0 (*e.g.* solve (IP2) for $t = 180s$),
- We try to improve this solution by a local search phase,

A Matheuristic for the GED problem

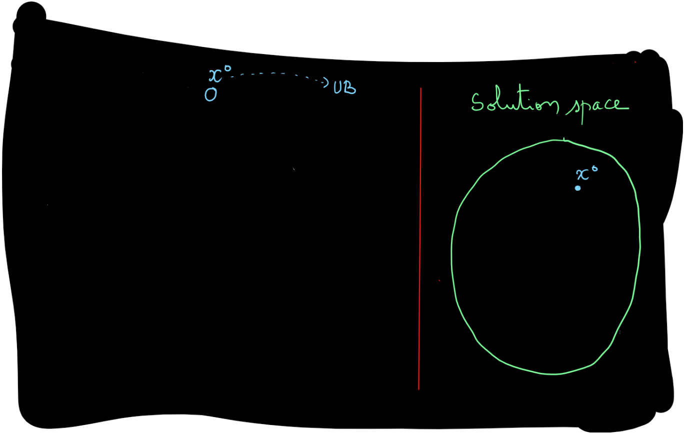
- We start with an initial solution x^0 (*e.g.* solve (IP2) for $t = 180$s),
- We try to improve this solution by a local search phase,
- Neighborhood definition $\mathcal{N}(x, x^\ell)$,
  $\mathcal{N}(x, x^\ell) = \{x/\Delta(x, x^\ell) \leq \pi\}$,
  with $\Delta(x, x^\ell) = \sum_{(i,k)\in S^\ell}(1 - x_{i,k}) + \sum_{(i,k)\notin S^\ell} x_{i,k}$,
  and $S^\ell = \{x_{u,v} \in x^\ell / x_{u,v} = 1\}$.

A Matheuristic for the GED problem

- We start with an initial solution x^0 (*e.g.* solve (IP2) for $t = 180$s),
- We try to improve this solution by a local search phase,
- Neighborhood definition $\mathcal{N}(x, x^\ell)$,
  $\mathcal{N}(x, x^\ell) = \{x / \Delta(x, x^\ell) \leq \pi\}$,
  with $\Delta(x, x^\ell) = \sum_{(i,k) \in S^\ell} (1 - x_{i,k}) + \sum_{(i,k) \notin S^\ell} x_{i,k}$,
  and $S^\ell = \{x_{u,v} \in x^\ell / x_{u,v} = 1\}$.
- Let us denote by $(\text{IP2})^I_\pi(x^\ell)$ the model (IP2) with the constraint $x \in \mathcal{N}(x, x^\ell)$ added,

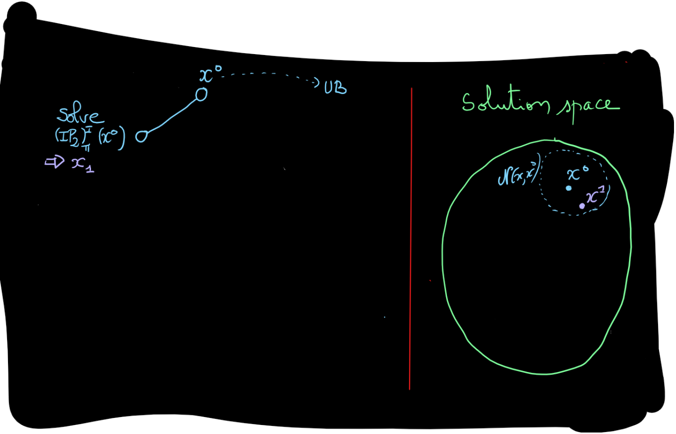
A Matheuristic for the GED problem

- Global functionning of the LocBra Matheuristic,



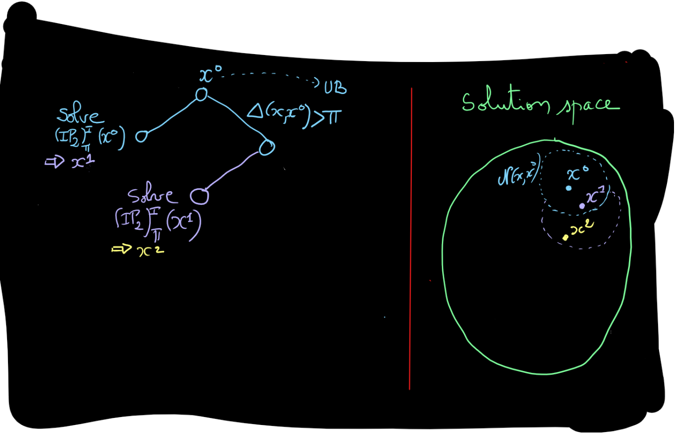
A Matheuristic for the GED problem

- Global functionning of the LocBra Matheuristic,



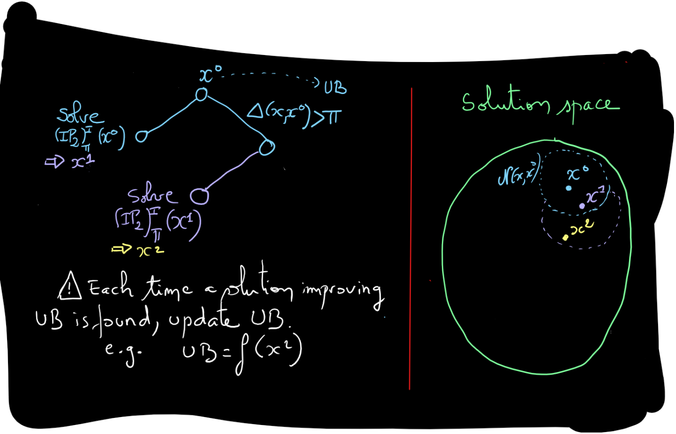
A Matheuristic for the GED problem

- Global functionning of the LocBra Matheuristic,



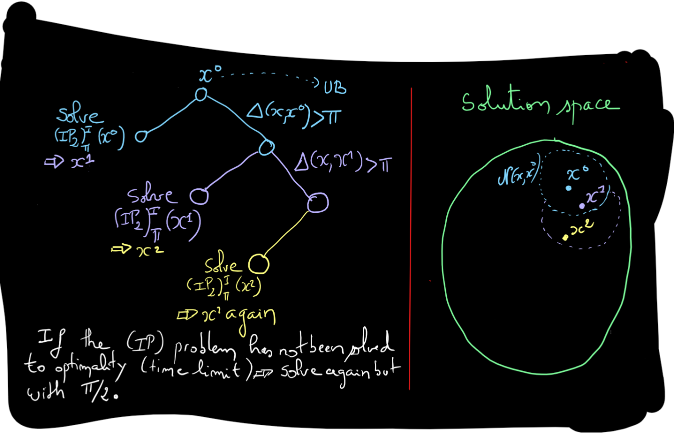
A Matheuristic for the GED problem

- Global functionning of the LocBra Matheuristic,



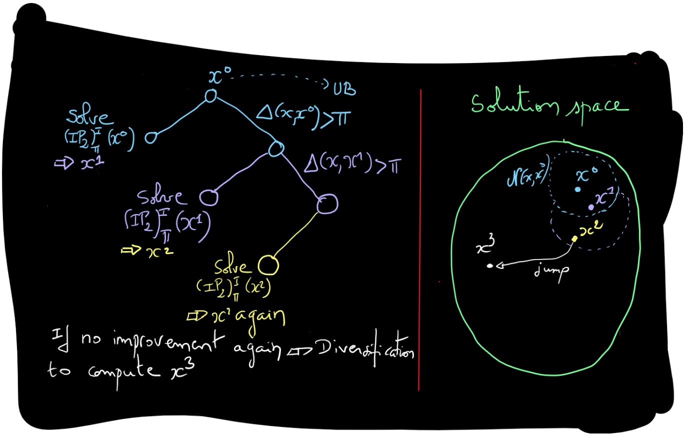
A Matheuristic for the GED problem

- Global functionning of the LocBra Matheuristic,



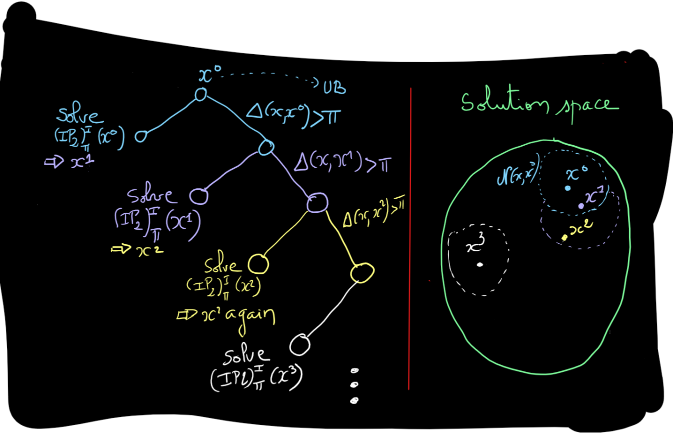
A Matheuristic for the GED problem

- Global functionning of the LocBra Matheuristic,



A Matheuristic for the GED problem

- Global functionning of the LocBra Matheuristic,



A Matheuristic for the GED problem

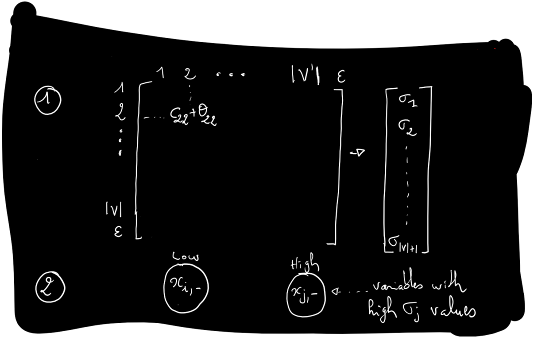
- Diversification,

A Matheuristic for the GED problem

- Diversification,
- Identify the variables $x_{i,k}$ which modification from the current solution x^{ℓ} implies a high modification of the objective function,

A Matheuristic for the GED problem

- Diversification,
- Identify the variables $x_{i,k}$ which modification from the current solution x^{ℓ} implies a high modification of the objective function,
- Compute costs $c_{i,k}$: cost of matching vertices $i \in V$ and $k \in V'$, Compute costs $\theta_{i,k}$: cost of matching edges from i with edges from k (assignment problem),



A Matheuristic for the GED problem

- Diversification,
- Let S_{div} be the set of variables $x_{i,k}$ with high σ_i values,

A Matheuristic for the GED problem

- Diversification,
- Let S_{div} be the set of variables $x_{i,k}$ with high σ_i values,
- To get a new solution from x^ℓ solve $(\text{IP2})^D_\beta(x^\ell)$:
  - Model $(IP2)$,
  - Add constraint:
    $\Delta'(x, x^\ell) = \left(\sum_{(i,k)\in S^\ell \cap S_{div}} (1 - x_{i,k}) + \sum_{(i,k)\in S_{div} \setminus S^\ell \cap S_{div}} x_{i,k} \right) \geq \beta$

Computational experiments

- How good is LocBra with respect to the state-of-the-art heuristics IPFP and GNCCP?

Computational experiments

- How good is LocBra with respect to the state-of-the-art heuristics IPFP and GNCCP?
- Several databases have been considered ([4]): MUTA, HOUSE-REF, PROTEIN,

Computational experiments

- How good is LocBra with respect to the state-of-the-art heuristics IPFP and GNCCP?
- Several databases have been considered ([4]): MUTA, HOUSE-REF, PROTEIN,
- Results on PROTEIN database,

Size	IPFP		GNCCP		LocBra	
	Avg Dev (%)	Avg time (s)	Avg Dev (%)	Avg time (s)	Avg Dev (%)	Avg time (s)
20x20	1.05	0.09	0.22	2.05	0.06	6.54
30x30	0.98	0.27	0.20	7.21	0.08	8.68
40x40	1.14	0.59	1.68	23.17	0.39	8.82

For graphs of size 20x20 and 30x30, we have the optimal solution. For 40x40 we have 63% of optimal solutions.

Computational experiments

- How good is LocBra with respect to the state-of-the-art heuristics IPFP and GNCCP?
- Several databases have been considered ([4]): MUTA, HOUSE-REF, PROTEIN,
- Results on PROTEIN database,

Size	IPFP		GNCCP		LocBra	
	Avg Dev (%)	Avg time (s)	Avg Dev (%)	Avg time (s)	Avg Dev (%)	Avg time (s)
20x20	1.05	0.09	0.22	2.05	0.06	6.54
30x30	0.98	0.27	0.20	7.21	0.08	8.68
40x40	1.14	0.59	1.68	23.17	0.39	8.82

  For graphs of size 20x20 and 30x30, we have the optimal solution. For 40x40 we have 63% of optimal solutions.
- A comparison with ground-truth on CMUHOUSE-NA shows that LocBra strongly outperforms the other heuristics (at most 5% of wrong matchings against more than 20% for the others).

Conclusions for the matheuristic

Cons

Conclusions for the matheuristic

Cons

- A lot of parameters to tune: (π, β, total_cpu_time, ub_cpu_time, ...),

Conclusions for the matheuristic

Cons

- A lot of parameters to tune: (π, β, total_cpu_time, ub_cpu_time, ...),
- Some parts (*e.g. diversification*) are efficient because problem dependent,

Conclusions for the matheuristic

Cons

- A lot of parameters to tune: (π, β, total_cpu_time, ub_cpu_time, ...),
- Some parts (*e.g. diversification*) are efficient because problem dependent,
- Need for a quite efficient MIP formulation.

Conclusions for the matheuristic

Cons

- A lot of parameters to tune: (π, β, total_cpu_time, ub_cpu_time, ...),
- Some parts (*e.g. diversification*) are efficient because problem dependent,
- Need for a quite efficient MIP formulation.

Pros

- The total CPU time allocated to the method can be tuned to fit user's requirements,

Conclusions for the matheuristic

Cons

- A lot of parameters to tune: (π, β, total_cpu_time, ub_cpu_time, ...),
- Some parts (*e.g. diversification*) are efficient because problem dependent,
- Need for a quite efficient MIP formulation.

Pros

- The total CPU time allocated to the method can be tuned to fit user's requirements,
- Very efficient and quite simple to use (black-box solver for the MIP),

Conclusions for the matheuristic

Cons

- A lot of parameters to tune: (π, β, total_cpu_time, ub_cpu_time, ...),
- Some parts (*e.g. diversification*) are efficient because problem dependent,
- Need for a quite efficient MIP formulation.

Pros

- The total CPU time allocated to the method can be tuned to fit user's requirements,
- Very efficient and quite simple to use (black-box solver for the MIP),
- Can be parallelized.

Conclusions

- Does Operations Research will save the world?

Conclusions

- Does Operations Research will save the world?
- No, but it can help to solve optimization/decision problems,

Conclusions

- Does Operations Research will save the world?
- No, but it can help to solve optimization/decision problems,
- Important issues:

Conclusions

- Does Operations Research will save the world?
- No, but it can help to solve optimization/decision problems,
- Important issues:
  - Modelling issues: having a good model to solve is **fundamental**,

Conclusions

- Does Operations Research will save the world?
- No, but it can help to solve optimization/decision problems,
- Important issues:
  - Modelling issues: having a good model to solve is **fundamental**,
  - Structural analysis: deriving mathematical properties is important to improve the solution,

Conclusions

- Does Operations Research will save the world?
- No, but it can help to solve optimization/decision problems,
- Important issues:
  - Modelling issues: having a good model to solve is **fundamental**,
  - Structural analysis: deriving mathematical properties is important to improve the solution,
  - Algorithmic issues: choose the right way (exact vs heuristic) and the appropriate algorithm.

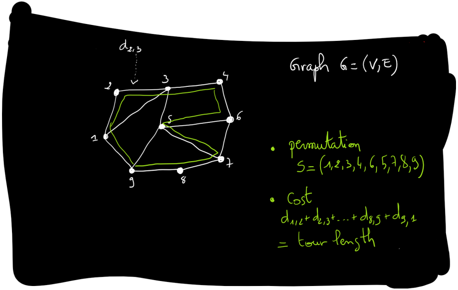
Beyond OR

- Some personal thoughts,

Beyond OR

- Some personal thoughts,
- Let us consider the **Traveling Salesman Problem** (TSP),
  Input: A connected graph $G = (V, E)$ with V the set of vertices (cities) and E the set of edges (routes). Each edge $(i, j) \in E$ is defined by a weight $d_{i,j}$ (distance). We note $n = |V|$.
  Goal: Find a permutation S of vertices such that

$$\left(\sum_{k=1}^{n} d_{S[k],S[k+1]} + d_{S[n],S[1]} \right) \text{ is minimum.}$$



The TSP

- What is the state-of-the-art in OR?

[10] Appelgate, D.L, Bixby, R.E., Chvatal, V. Cook, W.J. (2007). *The Traveling Salesman Problem: A Computational Study*, Princeton Press.

The TSP

- What is the state-of-the-art in OR?
- The problem is strongly $\mathcal{NP}$-hard,

[10] Appelgate, D.L, Bixby, R.E., Chvatal, V. Cook, W.J. (2007). *The Traveling Salesman Problem: A Computational Study*, Princeton Press.

The TSP

- What is the state-of-the-art in OR?
- The problem is strongly $\mathcal{NP}$-hard,
- Well solved to optimality by the Concorde solver ([10]) $\Rightarrow$ instances up to 86 000 cities,

[10] Appelgate, D.L, Bixby, R.E., Chvatal, V. Cook, W.J. (2007). *The Traveling Salesman Problem: A Computational Study*, Princeton Press.

The TSP

- What is the state-of-the-art in OR?
- The problem is strongly $\mathcal{NP}$-hard,
- Well solved to optimality by the Concorde solver ([10]) $\Rightarrow$ instances up to 86 000 cities,
- Based on a Branch-and-Cut algorithm exploiting mathematical programming,

[10] Appelgate, D.L, Bixby, R.E., Chvatal, V. Cook, W.J. (2007). *The Traveling Salesman Problem: A Computational Study*, Princeton Press.

The TSP

- A lot of heuristics exist: one of the most efficient one is LKH ([11,12,13]),

[11] Helsgaun, K. (2000). *An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic*, European Journal of Operational Research, 126(1):106-130.

[12] Helsgaun, K. (2009). *General k-opt submoves for the Lin-Kernighan TSP heuristic*, Mathematical Programming Computation, 1(2-3):119-163.

[13] Tinos, R., Helsgaun, K., Whitley, D. (2018). *Efficient Recombination in the Lin-Kernighan-Helsgaun Traveling Salesman Heuristic*, PPSN XV, pp. 95-107.

The TSP

- A lot of heuristics exist: one of the most efficient one is LKH ([11,12,13]),
- A local search heuristic using $2k$-opt as an operator to build a new solution: for any $k \geq 1$ value, k edges of the current solution are removed and k edges are added,

[11] Helsgaun, K. (2000). *An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic*, European Journal of Operational Research, 126(1):106-130.

[12] Helsgaun, K. (2009). *General k-opt submoves for the Lin-Kernighan TSP heuristic*, Mathematical Programming Computation, 1(2-3):119-163.

[13] Tinos, R., Helsgaun, K., Whitley, D. (2018). *Efficient Recombination in the Lin-Kernighan-Helsgaun Traveling Salesman Heuristic*, PPSN XV, pp. 95-107.

The TSP

- A lot of heuristics exist: one of the most efficient one is LKH ([11,12,13]),
- A local search heuristic using $2k$-opt as an operator to build a new solution: for any $k \geq 1$ value, k edges of the current solution are removed and k edges are added,
- Initial solution: random,

[11] Helsgaun, K. (2000). *An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic*, European Journal of Operational Research, 126(1):106-130.

[12] Helsgaun, K. (2009). *General k-opt submoves for the Lin-Kernighan TSP heuristic*, Mathematical Programming Computation, 1(2-3):119-163.

[13] Tinos, R., Helsgaun, K., Whitley, D. (2018). *Efficient Recombination in the Lin-Kernighan-Helsgaun Traveling Salesman Heuristic*, PPSN XV, pp. 95-107.

The TSP

- A lot of heuristics exist: one of the most efficient one is LKH ([11,12,13]),
- A local search heuristic using $2k$-opt as an operator to build a new solution: for any $k \geq 1$ value, k edges of the current solution are removed and k edges are added,
- Initial solution: random,
- Often finds optimal solutions, at most at 0.162% of the optimal solution (TSPlib),

[11] Helsgaun, K. (2000). *An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic*, European Journal of Operational Research, 126(1):106-130.

[12] Helsgaun, K. (2009). *General k-opt submoves for the Lin-Kernighan TSP heuristic*, Mathematical Programming Computation, 1(2-3):119-163.

[13] Tinos, R., Helsgaun, K., Whitley, D. (2018). *Efficient Recombination in the Lin-Kernighan-Helsgaun Traveling Salesman Heuristic*, PPSN XV, pp. 95-107.

The TSP

- A lot of heuristics exist: one of the most efficient one is LKH ([11,12,13]),
- A local search heuristic using $2k$-opt as an operator to build a new solution: for any $k \geq 1$ value, k edges of the current solution are removed and k edges are added,
- Initial solution: random,
- Often finds optimal solutions, at most at 0.162% of the optimal solution (TSPlib),
- Time complexity in $O(n^{2.2})$: instance with 13509 cities $\approx$ 12h.

[11] Helsgaun, K. (2000). *An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic*, European Journal of Operational Research, 126(1):106-130.

[12] Helsgaun, K. (2009). *General k-opt submoves for the Lin-Kernighan TSP heuristic*, Mathematical Programming Computation, 1(2-3):119-163.

[13] Tinos, R., Helsgaun, K., Whitley, D. (2018). *Efficient Recombination in the Lin-Kernighan-Helsgaun Traveling Salesman Heuristic*, PPSN XV, pp. 95-107.

The TSP

- A well solved problem today,

[14] Vinyals, O., Fortunato, M., Jaitly, N. (2015) *Pointer Networks*, in C. Cortes and N. D. Lawrence and D. D. Lee and M. Sugiyama and R. Garnett (Eds): Advances in Neural Information Processing Systems, 28:2692-2700.

The TSP

- A well solved problem today,
- It has started to be studied in ML,

[14] Vinyals, O., Fortunato, M., Jaitly, N. (2015) *Pointer Networks*, in C. Cortes and N. D. Lawrence and D. D. Lee and M. Sugiyama and R. Garnett (Eds): Advances in Neural Information Processing Systems, 28:2692-2700.

The TSP

- A well solved problem today,
- It has started to be studied in ML,
- Why?

[14] Vinyals, O., Fortunato, M., Jaitly, N. (2015) *Pointer Networks*, in C. Cortes and N. D. Lawrence and D. D. Lee and M. Sugiyama and R. Garnett (Eds): Advances in Neural Information Processing Systems, 28:2692-2700.

The TSP

- A well solved problem today,
- It has started to be studied in ML,
- Why?
- To get a very fast heuristic finding near-optimal solutions (the Holy Grail),

[14] Vinyals, O., Fortunato, M., Jaitly, N. (2015) *Pointer Networks*, in C. Cortes and N. D. Lawrence and D. D. Lee and M. Sugiyama and R. Garnett (Eds): Advances in Neural Information Processing Systems, 28:2692-2700.

The TSP

- A well solved problem today,
- It has started to be studied in ML,
- Why?
- To get a very fast heuristic finding near-optimal solutions (the Holy Grail),
- Let us consider the Pointer Network approach in [14],

[14] Vinyals, O., Fortunato, M., Jaitly, N. (2015) *Pointer Networks*, in C. Cortes and N. D. Lawrence and D. D. Lee and M. Sugiyama and R. Garnett (Eds): Advances in Neural Information Processing Systems, 28:2692-2700.

The TSP

- A well solved problem today,
- It has started to be studied in ML,
- Why?
- To get a very fast heuristic finding near-optimal solutions (the Holy Grail),
- Let us consider the Pointer Network approach in [14],
- Learn how to produce good solutions directly from the TSP instance,

[14] Vinyals, O., Fortunato, M., Jaitly, N. (2015) *Pointer Networks*, in C. Cortes and N. D. Lawrence and D. D. Lee and M. Sugiyama and R. Garnett (Eds): Advances in Neural Information Processing Systems, 28:2692-2700.

The TSP

- Training:

The TSP

- Training:
  - for instances with less than 20 cities, solve the problem to optimality (Held-Karp algorithm),

The TSP

- Training:
  - for instances with less than 20 cities, solve the problem to optimality (Held-Karp algorithm),
  - for instances with 20-50 cities, use Christofides heuristic,

The TSP

- Training:
  - for instances with less than 20 cities, solve the problem to optimality (Held-Karp algorithm),
  - for instances with 20-50 cities, use Christofides heuristic,
- Architecture: Pointer Network (RNN with an attention mechanism),

The TSP

- Training:
  - for instances with less than 20 cities, solve the problem to optimality (Held-Karp algorithm),
  - for instances with 20-50 cities, use Christofides heuristic,
- Architecture: Pointer Network (RNN with an attention mechanism),
- Predictor: Input=list of cities, Output=permutation,

The TSP

- Training:
  - for instances with less than 20 cities, solve the problem to optimality (Held-Karp algorithm),
  - for instances with 20-50 cities, use Christofides heuristic,
- Architecture: Pointer Network (RNN with an attention mechanism),
- Predictor: Input=list of cities, Output=permutation,
- So, in fact, the trained Pointer Network mimics existing (but not the most efficient ones) heuristics,

The TSP

- Training:
  - for instances with less than 20 cities, solve the problem to optimality (Held-Karp algorithm),
  - for instances with 20-50 cities, use Christofides heuristic,
- Architecture: Pointer Network (RNN with an attention mechanism),
- Predictor: Input=list of cities, Output=permutation,
- So, in fact, the trained Pointer Network mimics existing (but not the most efficient ones) heuristics,
- Result: it is worse than the Christofides heuristic,

The TSP

- Training:
  - for instances with less than 20 cities, solve the problem to optimality (Held-Karp algorithm),
  - for instances with 20-50 cities, use Christofides heuristic,
- Architecture: Pointer Network (RNN with an attention mechanism),
- Predictor: Input=list of cities, Output=permutation,
- So, in fact, the trained Pointer Network mimics existing (but not the most efficient ones) heuristics,
- Result: it is worse than the Christofides heuristic,
- Complexity of the predictor: $O(n^2)$... almost the same than LKH heuristic.

The TSP

- The same “problem” holds for other ML approaches,

The TSP

- The same "problem" holds for other ML approaches,
- How can we expect to outperform existing heuristics when we learn from... these heuristics?

The TSP

- The same “problem” holds for other ML approaches,
- How can we expect to outperform existing heuristics when we learn from... these heuristics?
- Having only optimal solutions in the training databases is difficult, so...

The TSP

- The same "problem" holds for other ML approaches,
- How can we expect to outperform existing heuristics when we learn from... these heuristics?
- Having only optimal solutions in the training databases is difficult, so...
- Is ML convicted to fail when applied to optimization problems?

The TSP

- The same “problem” holds for other ML approaches,
- How can we expect to outperform existing heuristics when we learn from... these heuristics?
- Having only optimal solutions in the training databases is difficult, so...
- Is ML convicted to fail when applied to optimization problems?
- The right question should be more **why** using ML,

OR and ML

- Two important reasons ([15]):

[15] Bengio, Y., Lodi, A., Prouvost, A. (2019) *Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon*, Research report, arXiv, arxiv.org/abs/1811.06128v1.

OR and ML

- Two important reasons ([15]):
  - To build an heuristic that is **fast**... and possibly mimics slow effective OR heuristics,

[15] Bengio, Y., Lodi, A., Prouvost, A. (2019) *Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon*, Research report, arXiv, arxiv.org/abs/1811.06128v1.

OR and ML

- Two important reasons ([15]):
  - To build an heuristic that is **fast**... and possibly mimics slow effective OR heuristics,
  - When a real-life optimization problem cannot be mathematically formalized in an acceptable way,

[15] Bengio, Y., Lodi, A., Prouvost, A. (2019) *Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon*, Research report, arXiv, arxiv.org/abs/1811.06128v1.

OR and ML

- Two important reasons ([15]):
  - To build an heuristic that is **fast**... and possibly mimics slow effective OR heuristics,
  - When a real-life optimization problem cannot be mathematically formalized in an acceptable way,
- In the remainder, consider only "well-defined" optimization problems,

[15] Bengio, Y., Lodi, A., Prouvost, A. (2019) *Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon*, Research report, arXiv, arxiv.org/abs/1811.06128v1.

OR and ML

- Two important reasons ([15]):
  - To build an heuristic that is **fast**... and possibly mimics slow effective OR heuristics,
  - When a real-life optimization problem cannot be mathematically formalized in an acceptable way,
- In the remainder, consider only "well-defined" optimization problems,
- My believe (also expressed somehow in [15]):

ML and OR should no longer be used separately to solve optimization problems.

[15] Bengio, Y., Lodi, A., Prouvost, A. (2019) *Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon*, Research report, arXiv, arxiv.org/abs/1811.06128v1.

OR and ML

- ML: elaborates on data / OR: elaborates on the constraints and objectives,

OR and ML

- ML: elaborates on data / OR: elaborates on the constraints and objectives,
- OR algorithms strongly exploit the structure of problems... but they are sometimes conceived on "pifométrique" rules,

OR and ML

- ML: elaborates on data / OR: elaborates on the constraints and objectives,
- OR algorithms strongly exploit the structure of problems... but they are sometimes conceived on "pifométrique" rules,
- The future? Embed ML into OR algorithms to remove these rules!

OR and ML

- ML: elaborates on data / OR: elaborates on the constraints and objectives,
- OR algorithms strongly exploit the structure of problems... but they are sometimes conceived on “pifométrique” rules,
- The future? Embed ML into OR algorithms to remove these rules!
- Let us go back to the LocBra heuristic for the GED problem,

OR and ML

- ML: elaborates on data / OR: elaborates on the constraints and objectives,
- OR algorithms strongly exploit the structure of problems... but they are sometimes conceived on "pifométrique" rules,
- The future? Embed ML into OR algorithms to remove these rules!
- Let us go back to the LocBra heuristic for the GED problem,
- The neighborhood definition used to do intensification is:

$$\Delta(x, x^{\ell}) \leq \pi.$$

OR and ML

- The neighborhood definition used to do intensification is:
$$\Delta(x, x^{\ell}) \leq \pi.$$
- We let the MIP solver exploring the neighborhood $\Rightarrow$ time consuming,

OR and ML

- The neighborhood definition used to do intensification is:

$$\Delta(x, x^{\ell}) \leq \pi.$$

- We let the MIP solver exploring the neighborhood $\Rightarrow$ time consuming,
- The "pifométrique rule": *As I don't know the interesting parts of the neighborhood, I pay for intensive computations*,

OR and ML

- The neighborhood definition used to do intensification is:

$$\Delta(x, x^{\ell}) \leq \pi.$$

- We let the MIP solver exploring the neighborhood $\Rightarrow$ time consuming,
- The "pifométrique rule": *As I don't know the interesting parts of the neighborhood, I pay for intensive computations*,
- Why not learning, for a given current solution x^{ℓ}, what is the most promising part of the neighborhood (subset of variables to consider)?

OR and ML

- The neighborhood definition used to do intensification is:
$$\Delta(x, x^\ell) \leq \pi.$$
- We let the MIP solver exploring the neighborhood $\Rightarrow$ time consuming,
- The "pifométrique rule": *As I don't know the interesting parts of the neighborhood, I pay for intensive computations*,
- Why not learning, for a given current solution x^ℓ, what is the most promising part of the neighborhood (subset of variables to consider)?
- This would make faster the intensification phase $\Rightarrow$ enable to consider larger neighborhoods $\Rightarrow$ improve the overall LocBra heuristic.

OR and ML

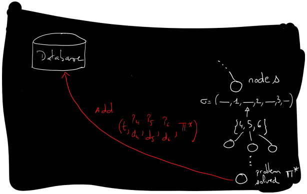
- And there are a very long list of possible integrations of ML into OR algorithms,

OR and ML

- And there are a very long list of possible integrations of ML into OR algorithms,
- Let us go back to the SMTT problem,

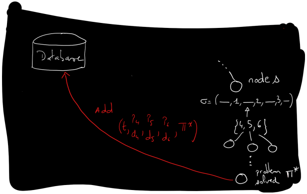
OR and ML

- And there are a very long list of possible integrations of ML into OR algorithms,
- Let us go back to the SMTT problem,
- We proposed to memorize an exponential number of solutions into a finite size database,



OR and ML

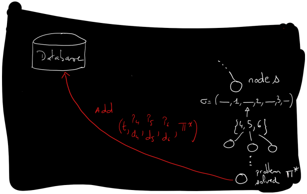
- And there are a very long list of possible integrations of ML into OR algorithms,
- Let us go back to the SMTT problem,
- We proposed to memorize an exponential number of solutions into a finite size database,



- Currently, the policy to update the database is: *remove all solutions that were never used to prune other nodes*,

OR and ML

- And there are a very long list of possible integrations of ML into OR algorithms,
- Let us go back to the SMTT problem,
- We proposed to memorize an exponential number of solutions into a finite size database,



- Currently, the policy to update the database is: *remove all solutions that were never used to prune other nodes*,
- Room for ML to learn if a solution will be dominant or not.

OR and ML

- Decades of exciting research activities to come!

OR and ML

- Decades of exciting research activities to come!
- ... and possibly, OR+ML may save the world.

OR and ML

- Decades of exciting research activities to come!
- ... and possibly, OR+ML may save the world.

Thank you for your attention.