A story about Machine Learning and Operations Research

Vincent T'kindt, Romain Raveaux and Jean-Paul Chemla

University of Tours,
LIFAT, France.

October 16, 2025

Outline

Outline

Purpose of this work

- Study a possible interaction between Machine Learning (ML) and Operations Research (OR),

Purpose of this work

- Study a possible interaction between Machine Learning (ML) and Operations Research (OR),
- Our believe: ML can help to improve algorithms from OR field to solve combinatorial optimization problems,

Purpose of this work

- Study a possible interaction between Machine Learning (ML) and Operations Research (OR),
- Our believe: ML can help to improve algorithms from OR field to solve combinatorial optimization problems,
- We have selected...
  - an $\mathcal{NP}$-hard scheduling problem,

Purpose of this work

- Study a possible interaction between Machine Learning (ML) and Operations Research (OR),
- Our believe: ML can help to improve algorithms from OR field to solve combinatorial optimization problems,
- We have selected...
  - an $\mathcal{NP}$-hard scheduling problem,
  - the best heuristic algorithm to solve it (a matheuristic),

Purpose of this work

- Study a possible interaction between Machine Learning (ML) and Operations Research (OR),
- Our believe: ML can help to improve algorithms from OR field to solve combinatorial optimization problems,
- We have selected...
  - an $\mathcal{NP}$-hard scheduling problem,
  - the best heuristic algorithm to solve it (a matheuristic),
- General question: How can ML lead to improve the matheuristic?

Purpose of this work

- Study a possible interaction between Machine Learning (ML) and Operations Research (OR),
- Our believe: ML can help to improve algorithms from OR field to solve combinatorial optimization problems,
- We have selected...
  - an $\mathcal{NP}$-hard scheduling problem,
  - the best heuristic algorithm to solve it (a matheuristic),
- General question: How can ML lead to improve the matheuristic?
  1. Decide of the "best" set of parameters,

Purpose of this work

- Study a possible interaction between Machine Learning (ML) and Operations Research (OR),
- Our believe: ML can help to improve algorithms from OR field to solve combinatorial optimization problems,
- We have selected...
  - an $\mathcal{NP}$-hard scheduling problem,
  - the best heuristic algorithm to solve it (a matheuristic),
- General question: How can ML lead to improve the matheuristic?
  1. Decide of the "best" set of parameters,
  2. Guide the matheuristic during its exploration of the search space.

State-of-the-art on ML for OR

- Coupling ML and OR is very trendy since the last years,

[10] Bengio, Y., Lodi, A., Prouvost, A. (2021). Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon. *European Journal of Operational Research*, 290(2):405-421.

State-of-the-art on ML for OR

- Coupling ML and OR is very trendy since the last years,
- A quick review of the literature (Bengio et al., [10]):

[10] Bengio, Y., Lodi, A., Prouvost, A. (2021). Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon. *European Journal of Operational Research*, 290(2):405-421.

State-of-the-art on ML for OR

- Coupling ML and OR is very trendy since the last years,
- A quick review of the literature (Bengio et al., [10]):
  1. Use ML in OR algorithms to replace time-consuming components,
     - Cutting plane selection (replacement of lower bound computation),
     - Selection of variables in a branching algorithm (replacement of strong branching strategy),
     - ...

[10] Bengio, Y., Lodi, A., Prouvost, A. (2021). Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon. *European Journal of Operational Research*, 290(2):405-421.

State-of-the-art on ML for OR

- Coupling ML and OR is very trendy since the last years,
- A quick review of the literature (Bengio et al., [10]):
  1. Use ML in OR algorithms to replace time-consuming components,
     - Cutting plane selection (replacement of lower bound computation),
     - Selection of variables in a branching algorithm (replacement of strong branching strategy),
     - ...
  2. Use ML to guide OR algorithms,
     - Learning parameters of algorithms,
     - To determine whether or not applying a Dantzig-Wolf decomposition or linearizing a quadratic problem is effective,
     - Node selection in branching algorithms (search strategy),
     - To decide of the call of heuristics at nodes in a branching algorithm,

[10] Bengio, Y., Lodi, A., Prouvost, A. (2021). Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon. *European Journal of Operational Research*, 290(2):405-421.

State-of-the-art on ML for OR

- Question 1: Is directly using ML to solve optimization problems a good idea?

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

State-of-the-art on ML for OR

- Question 1: Is directly using ML to solve optimization problems a good idea?
  - Usually, a predictor that builds a solution is trained from known heuristics: few chance to perform better than the original heuristics (e.g. [11] on the TSP: same CPU time and worse heuristic),

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

State-of-the-art on ML for OR

- Question 1: Is directly using ML to solve optimization problems a good idea?
  - Usually, a predictor that builds a solution is trained from known heuristics: few chance to perform better than the original heuristics (e.g. [11] on the TSP: same CPU time and worse heuristic),
  - Worthwhile if the original heuristics are very time consuming $\Rightarrow$ design of fast heuristics.

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

State-of-the-art on ML for OR

- Question 2: Is learning the search strategy or branching strategy a good idea?

[12] Alvarez A., Louveaux, Q., Wehenkel, L. (2017). A machine learning-based approximation of strong branching *INFORMS Journal on Computing*, 29(1):185-195.

State-of-the-art on ML for OR

- Question 2: Is learning the search strategy or branching strategy a good idea?
  - From practice we now that, on hard instances, an exact branching algorithm spends most of its CPU time (e.g. 80%) proving the optimality,

[12] Alvarez A., Louveaux, Q., Wehenkel, L. (2017). A machine learning-based approximation of strong branching *INFORMS Journal on Computing*, 29(1):185-195.

State-of-the-art on ML for OR

- Question 2: Is learning the search strategy or branching strategy a good idea?
  - From practice we now that, on hard instances, an exact branching algorithm spends most of its CPU time (e.g. 80%) proving the optimality,
  - Limited impact on the efficiency of an exact method (e.g. [12] and the case of strong branching: no real gain on the size or number of problems solved),

[12] Alvarez A., Louveaux, Q., Wehenkel, L. (2017). A machine learning-based approximation of strong branching *INFORMS Journal on Computing*, 29(1):185-195.

State-of-the-art on ML for OR

- Question 2: Is learning the search strategy or branching strategy a good idea?
  - From practice we now that, on hard instances, an exact branching algorithm spends most of its CPU time (e.g. 80%) proving the optimality,
  - Limited impact on the efficiency of an exact method (e.g. [12] and the case of strong branching: no real gain on the size or number of problems solved),
  - Worthwhile for search tree based heuristics,

[12] Alvarez A., Louveaux, Q., Wehenkel, L. (2017). A machine learning-based approximation of strong branching *INFORMS Journal on Computing*, 29(1):185-195.

State-of-the-art on ML for OR

- Question 3: Why integrating ML into OR algorithms?

State-of-the-art on ML for OR

- Question 3: Why integrating ML into OR algorithms?
  1. Exact algorithms: to enable them to solve larger problem instances.

State-of-the-art on ML for OR

- Question 3: Why integrating ML into OR algorithms?
  1. Exact algorithms: to enable them to solve larger problem instances.
  2. Heuristic algorithms: to create fast or more efficient heuristics.

State-of-the-art on ML for OR

- Question 3: Why integrating ML into OR algorithms?
  1. Exact algorithms: to enable them to solve larger problem instances.
  2. Heuristic algorithms: to create fast or more efficient heuristics.
- We focus on the second aspect and notably consider two research directions:

State-of-the-art on ML for OR

- Question 3: Why integrating ML into OR algorithms?
  1. Exact algorithms: to enable them to solve larger problem instances.
  2. Heuristic algorithms: to create fast or more efficient heuristics.
- We focus on the second aspect and notably consider two research directions:
  1. How to optimize the parameters of a heuristic?

State-of-the-art on ML for OR

- Question 3: Why integrating ML into OR algorithms?
  1. Exact algorithms: to enable them to solve larger problem instances.
  2. Heuristic algorithms: to create fast or more efficient heuristics.
- We focus on the second aspect and notably consider two research directions:
  1. How to optimize the parameters of a heuristic?
  2. How to guide a local search heuristic?

State-of-the-art on ML for OR

- Question 3: Why integrating ML into OR algorithms?
  1. Exact algorithms: to enable them to solve larger problem instances.
  2. Heuristic algorithms: to create fast or more efficient heuristics.
- We focus on the second aspect and notably consider two research directions:
  1. How to optimize the parameters of a heuristic?
  2. How to guide a local search heuristic?
- We provide some answers to these two questions for a particular scheduling problem and a matheuristic algorithm.

The scheduling problem

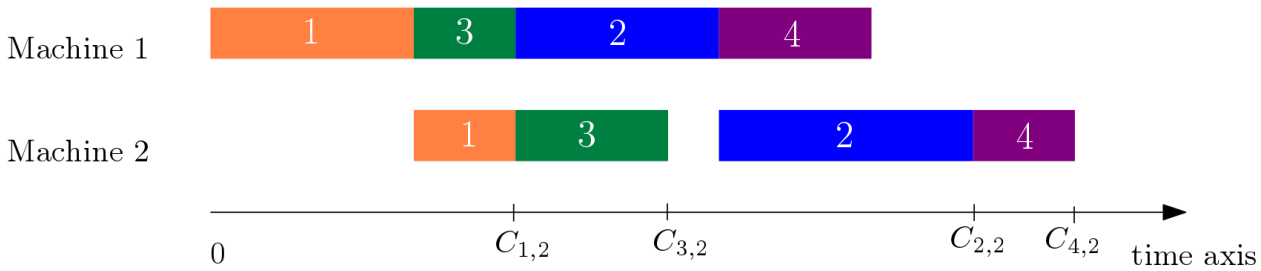
- n jobs have to be scheduled on two machines,

The scheduling problem

- n jobs have to be scheduled on two machines,
- Each job j is defined by a *processing time* $p_{j,i}$ on machine $i = 1, 2$,

The scheduling problem

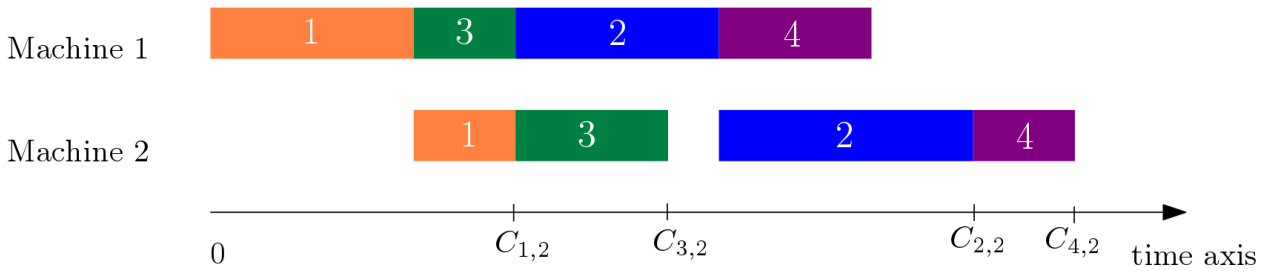
- n jobs have to be scheduled on two machines,
- Each job j is defined by a *processing time* $p_{j,i}$ on machine $i = 1, 2$,
- Machines are organized in a flowshop setting: each job has to be processed first on machine 1 and next on machine 2,



$C_{j,i}$: completion time of job j on machine i,

The scheduling problem

- n jobs have to be scheduled on two machines,
- Each job j is defined by a *processing time* $p_{j,i}$ on machine $i = 1, 2$,
- Machines are organized in a flowshop setting: each job has to be processed first on machine 1 and next on machine 2,

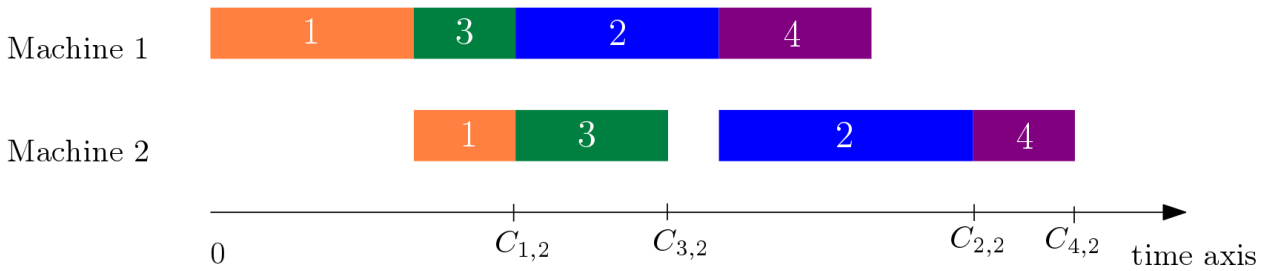


$C_{j,i}$: completion time of job j on machine i,

- A schedule is a permutation σ of the jobs,

The scheduling problem

- n jobs have to be scheduled on two machines,
- Each job j is defined by a *processing time* $p_{j,i}$ on machine $i = 1, 2$,
- Machines are organized in a flowshop setting: each job has to be processed first on machine 1 and next on machine 2,

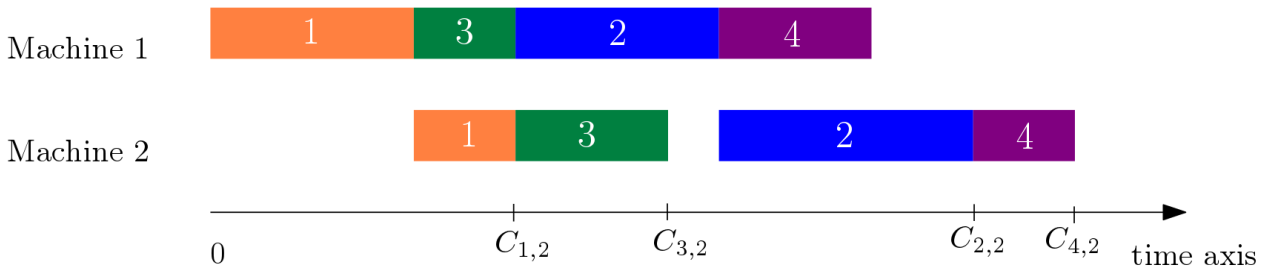


$C_{j,i}$: completion time of job j on machine i,

- A schedule is a permutation σ of the jobs,
- A schedule is optimal if it minimizes $\sum_j C_{j,2}$,

The scheduling problem

- n jobs have to be scheduled on two machines,
- Each job j is defined by a *processing time* $p_{j,i}$ on machine $i = 1, 2$,
- Machines are organized in a flowshop setting: each job has to be processed first on machine 1 and next on machine 2,



$C_{j,i}$: completion time of job j on machine i,

- A schedule is a permutation σ of the jobs,
- A schedule is optimal if it minimizes $\sum_j C_{j,2}$,

$\Rightarrow$ The $F2||\sum_j C_{j,2}$ problem is strongly $\mathcal{NP}$-hard.

The scheduling problem

- A well studied problem in the literature,

[1] Detienne, B., Sadykov, R., Tanaka, S. (2016). The two-machine flowshop total completion time problem: Branch-and-bound algorithms based on network-flow formulation. *European Journal of Operational Research*, 252(3):750-760.
[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annal of Operations Research*, 213:67-78.

The scheduling problem

- A well studied problem in the literature,
- Best exact algorithm: a branch-and-bound algorithm proposed by Detienne, Sadykov and Tanaka ([1]) $\Rightarrow$ solve up to 100-job instances,

[1] Detienne, B., Sadykov, R., Tanaka, S. (2016). The two-machine flowshop total completion time problem: Branch-and-bound algorithms based on network-flow formulation. *European Journal of Operational Research*, 252(3):750-760.
[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annal of Operations Research*, 213:67-78.

The scheduling problem

- A well studied problem in the literature,
- Best exact algorithm: a branch-and-bound algorithm proposed by Detienne, Sadykov and Tanaka ([1]) $\Rightarrow$ solve up to 100-job instances,
- Best heuristic: a matheuristic (`MATH`) proposed by Della Croce, Grosso and Salassa ([2]),

[1] Detienne, B., Sadykov, R., Tanaka, S. (2016). The two-machine flowshop total completion time problem: Branch-and-bound algorithms based on network-flow formulation. *European Journal of Operational Research*, 252(3):750-760.
[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annal of Operations Research*, 213:67-78.

The scheduling problem

- A well studied problem in the literature,
- Best exact algorithm: a branch-and-bound algorithm proposed by Detienne, Sadykov and Tanaka ([1]) $\Rightarrow$ solve up to 100-job instances,
- Best heuristic: a matheuristic (`MATH`) proposed by Della Croce, Grosso and Salassa ([2]),
  - Provide near-optimal solutions: for $n = 300$, GapToOptimality$< 0.1\%$,
  - CPU times required $= 300$ seconds.

[1] Detienne, B., Sadykov, R., Tanaka, S. (2016). The two-machine flowshop total completion time problem: Branch-and-bound algorithms based on network-flow formulation. *European Journal of Operational Research*, 252(3):750-760.
[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annal of Operations Research*, 213:67-78.

The scheduling problem

- A well studied problem in the literature,
- Best exact algorithm: a branch-and-bound algorithm proposed by Detienne, Sadykov and Tanaka ([1]) $\Rightarrow$ solve up to 100-job instances,
- Best heuristic: a matheuristic (MATH) proposed by Della Croce, Grosso and Salassa ([2]),
  - Provide near-optimal solutions: for $n = 300$, GapToOptimality$< 0.1\%$,
  - CPU times required $= 300$ seconds.
- Our goal: integrate ML techniques to improve the matheuristic (ℓ-MATH).

[1] Detienne, B., Sadykov, R., Tanaka, S. (2016). The two-machine flowshop total completion time problem: Branch-and-bound algorithms based on network-flow formulation. *European Journal of Operational Research*, 252(3):750-760.
[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annal of Operations Research*, 213:67-78.

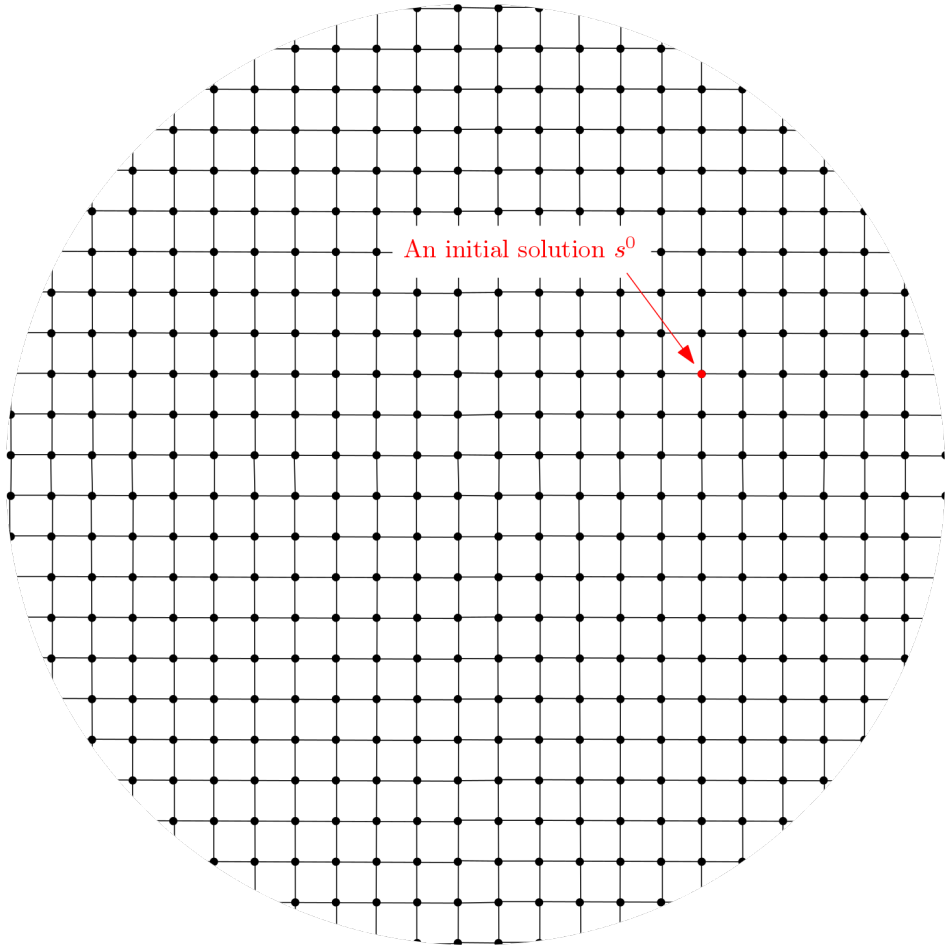
The MATH heuristic

- Matheuristic as *LNS* heuristics ([1,3]),

[1] Fischetti, M., Fischetti, M. (2018). Matheuristics. In: Martí R., Pardalos P., Resende M. (eds), Handbook of Heuristics. *Springer*.
[3] Della Croce, F., A. Grosso, F. Salassa (2013). Matheuristics: Embedding MILP solvers into heuristic algorithms for combinatorial optimization problems, P. Siarry (Eds): Heuristics: Theory and Application, *Nova Science Publisher*, 53–68.

The MATH heuristic

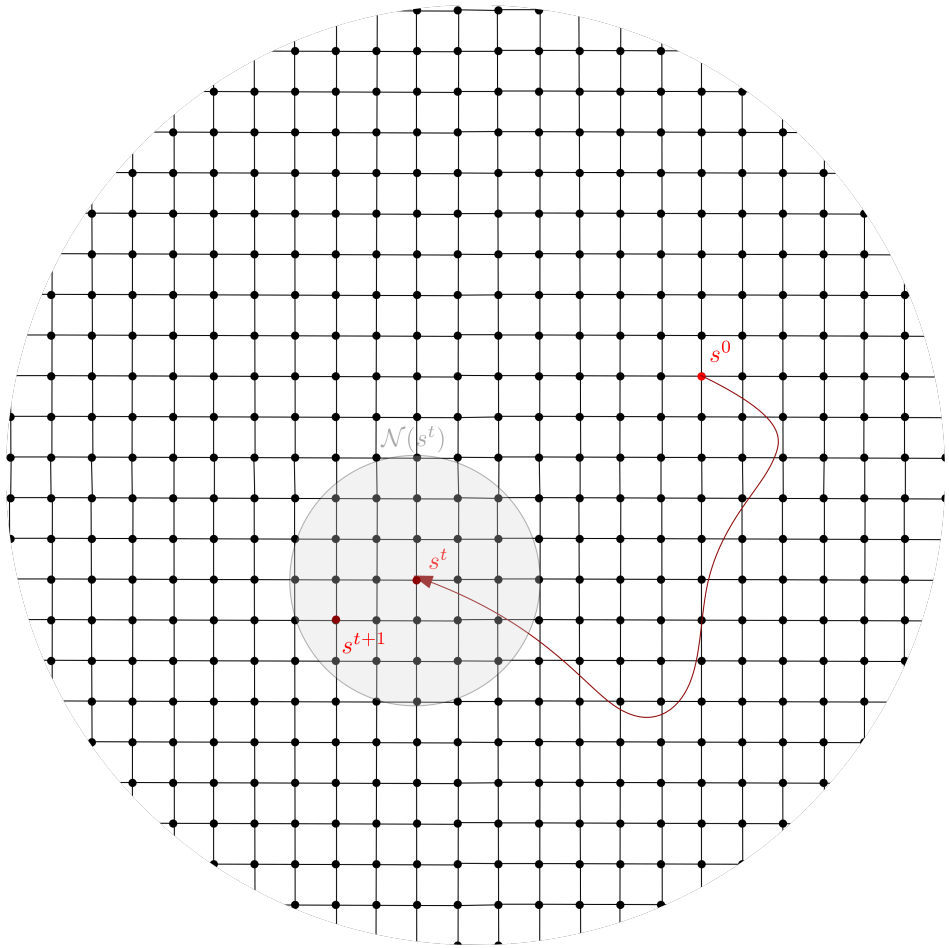
- Matheuristic as *LNS* heuristics ([1,3]),
- We start from a known initial solution s^0,



[1] Fischetti, M., Fischetti, M. (2018). Matheuristics. In: Martí R., Pardalos P., Resende M. (eds), Handbook of Heuristics. *Springer*.
[3] Della Croce, F., A. Grosso, F. Salassa (2013). Matheuristics: Embedding MILP solvers into heuristic algorithms for combinatorial optimization problems, P. Siarry (Eds): Heuristics: Theory and Application, *Nova Science Publisher*, 53–68.

The MATH heuristic

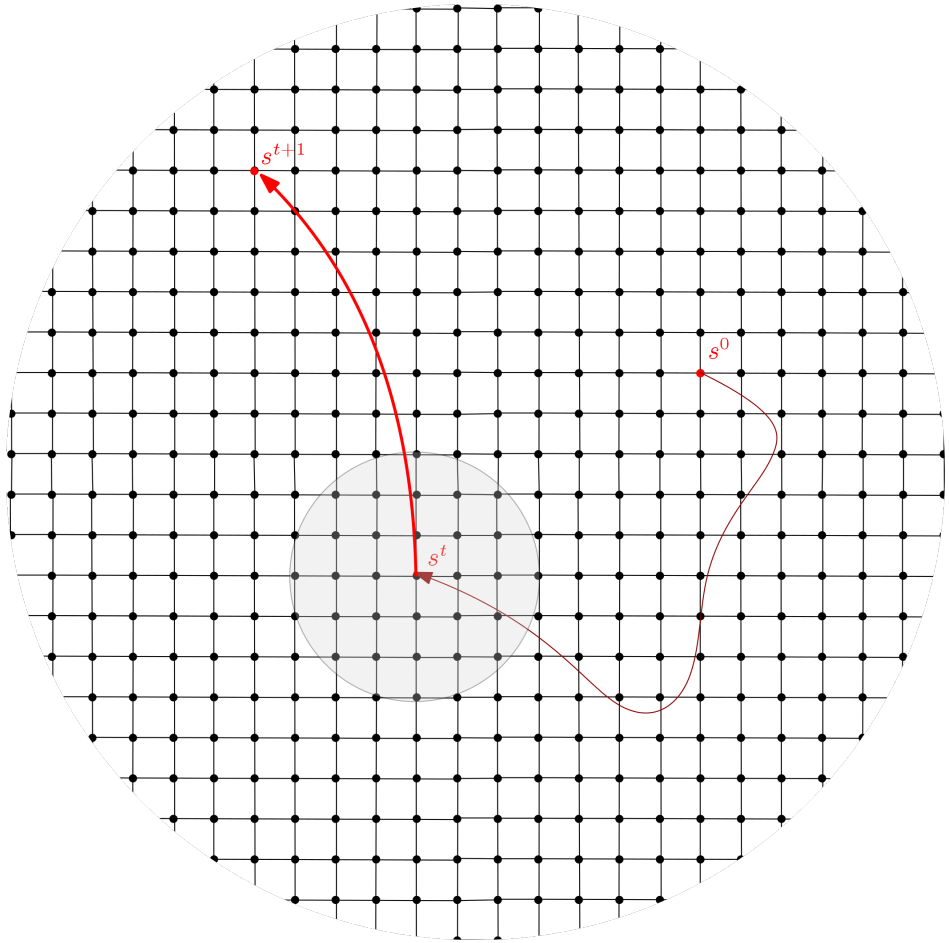
- Matheuristic as *LNS* heuristics ([1,3]),
- We start from a known initial solution s^0,
- Exploration of the neighbourhood $\mathcal{N}(s^t)$: by MIP (*intensification*),



[1] Fischetti, M., Fischetti, M. (2018). Matheuristics. In: Martí R., Pardalos P., Resende M. (eds), Handbook of Heuristics. *Springer*.
[3] Della Croce, F., A. Grosso, F. Salassa (2013). Matheuristics: Embedding MILP solvers into heuristic algorithms for combinatorial optimization problems, P. Siarry (Eds): Heuristics: Theory and Application, *Nova Science Publisher*, 53–68.

The MATH heuristic

- Matheuristic as *LNS* heuristics ([1,3]),
- We start from a known initial solution s^0,
- Exploration of the neighbourhood $\mathcal{N}(s^t)$: by MIP (*intensification*),
- In case of local optimum: *diversification* by MIP.



[1] Fischetti, M., Fischetti, M. (2018). Matheuristics. In: Martí R., Pardalos P., Resende M. (eds), Handbook of Heuristics. *Springer*.
[3] Della Croce, F., A. Grosso, F. Salassa (2013). Matheuristics: Embedding MILP solvers into heuristic algorithms for combinatorial optimization problems, P. Siarry (Eds): Heuristics: Theory and Application, *Nova Science Publisher*, 53–68.

VPLS: the recipe

- Exploit a direct *position-based* IP formulation: $x_{ij} = 1$ is job j is in position i; 0 otherwise,

[5] Della Croce, F., Ghirardi, M., Tadei, R. (2004). Recovering Beam Search: enhancing the beam search approach for combinatorial optimization problems. *Journal of Heuristics*, 10:89-104.

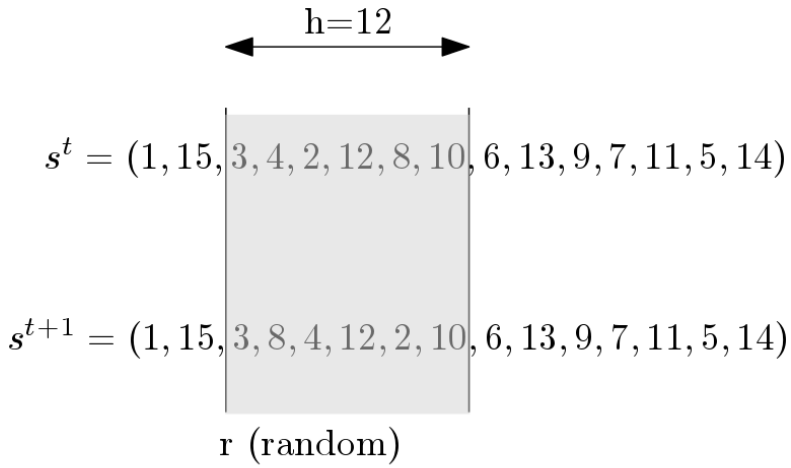
VPLS: the recipe

- Exploit a direct *position-based* IP formulation: $x_{ij} = 1$ is job j is in position i; 0 otherwise,
- `Initialization:` s^0 is computed by the *Recovering Beam Search* heuristic of [5],

[5] Della Croce, F., Ghirardi, M., Tadei, R. (2004). Recovering Beam Search: enhancing the beam search approach for combinatorial optimization problems. *Journal of Heuristics*, 10:89-104.

VPLS: the recipe

- Exploit a direct *position-based* IP formulation: $x_{ij} = 1$ is job j is in position i; 0 otherwise,
- `Initialization:` s^0 is computed by the *Recovering Beam Search* heuristic of [5],
- `Neighbourhood definition` $\mathcal{N}(s^t)$:

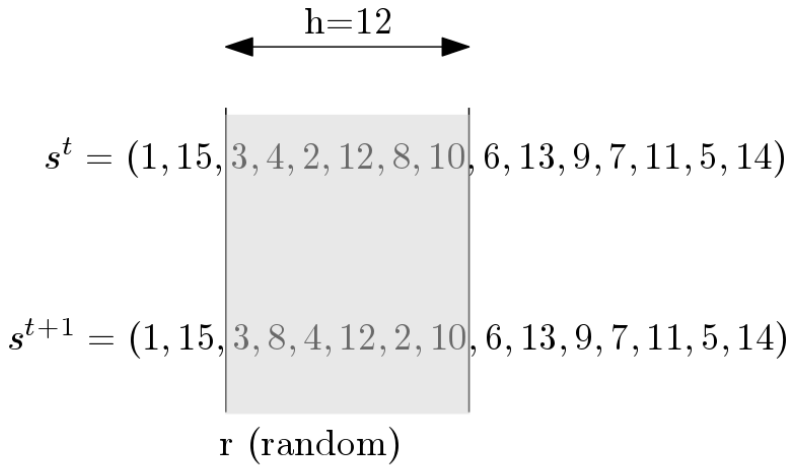


$$\mathcal{S}^t = \{x_{ij} | i = 1..r-1, r+h+1, ...n, j = 1..n\}$$

[5] Della Croce, F., Ghirardi, M., Tadei, R. (2004). Recovering Beam Search: enhancing the beam search approach for combinatorial optimization problems. *Journal of Heuristics*, 10:89-104.

VPLS: the recipe

- Exploit a direct *position-based* IP formulation: $x_{ij} = 1$ is job j is in position i; 0 otherwise,
- `Initialization:` s^0 is computed by the *Recovering Beam Search* heuristic of [5],
- `Neighbourhood definition` $\mathcal{N}(s^t)$:



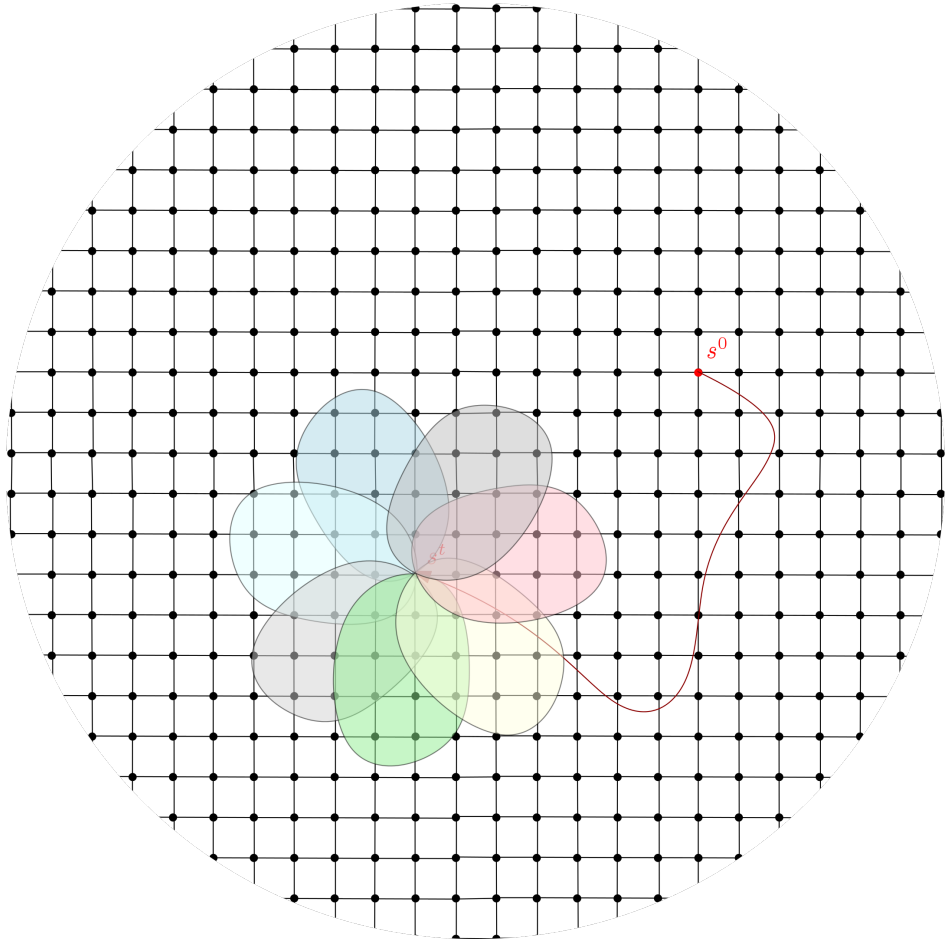
$$\mathcal{S}^t = \{x_{ij} | i = 1..r-1, r+h+1, ...n, j = 1..n\}$$

$\Rightarrow$ well suited for permutation problems.

[5] Della Croce, F., Ghirardi, M., Tadei, R. (2004). Recovering Beam Search: enhancing the beam search approach for combinatorial optimization problems. *Journal of Heuristics*, 10:89-104.

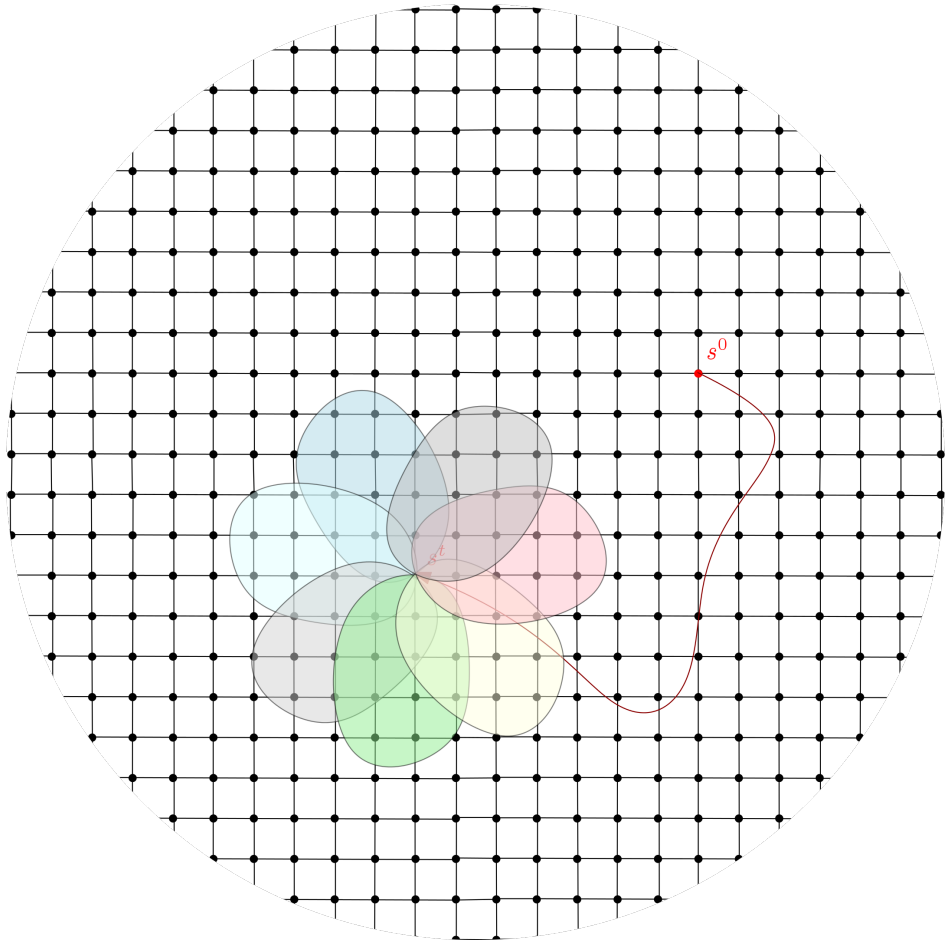
VPLS: the recipe

- Random selection of $r \Leftrightarrow$ random selection of $\mathcal{N}(s^t)$,



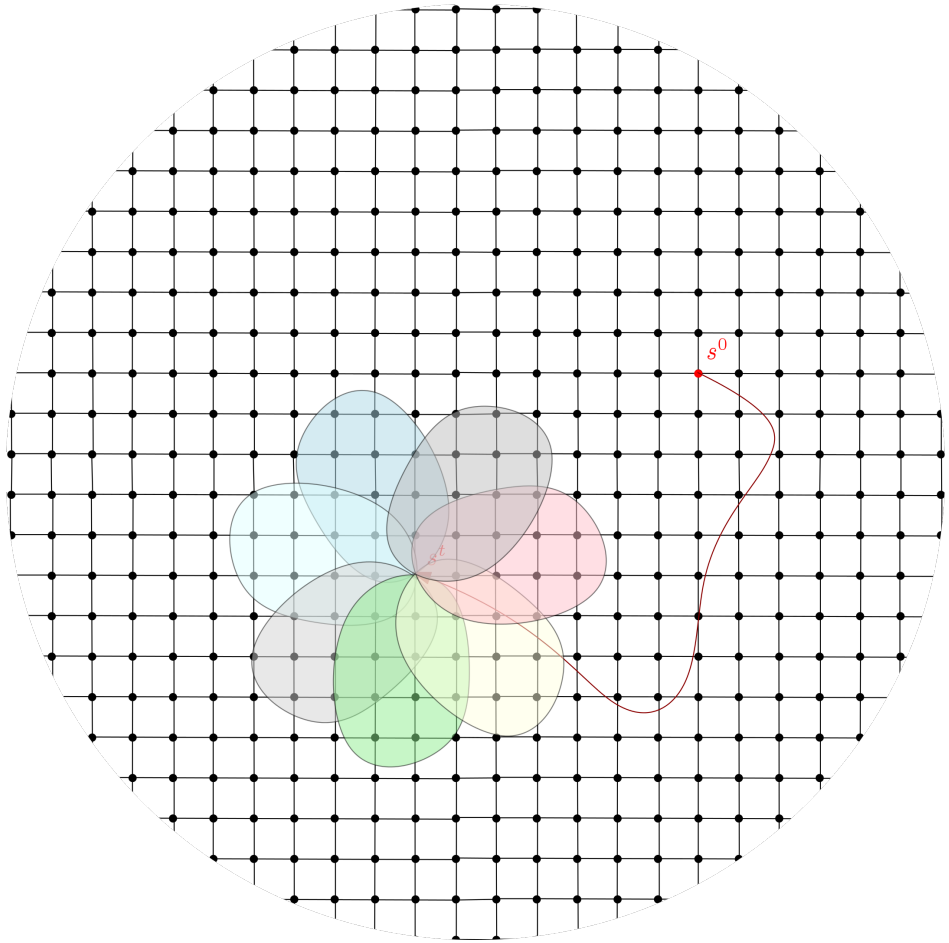
VPLS: the recipe

- Random selection of $r \Leftrightarrow$ random selection of $\mathcal{N}(s^t)$,
- First improving neighbourhood,



VPLS: the recipe

- Random selection of $r \Leftrightarrow$ random selection of $\mathcal{N}(s^t)$,
- First improving neighbourhood,
- `Stopping condition:` a given time limit T_{stop} is reached or no improving neighbourhood.



Outline

First questionning

- The VPLS matheuristic requests fixing several parameters:

First questionning

- The VPLS matheuristic requests fixing several parameters:
  1. The window size h,

First questionning

- The VPLS matheuristic requests fixing several parameters:
  1. The window size h,
  2. The total allowed CPU time T_{stop},

First questionning

- The VPLS matheuristic requests fixing several parameters:
  1. The window size h,
  2. The total allowed CPU time T_{stop},
  3. The allowed CPU time for exploring a neighbourhood T_{iter}.

First questionning

- The VPLS matheuristic requests fixing several parameters:
  1. The window size h,
  2. The total allowed CPU time T_{stop},
  3. The allowed CPU time for exploring a neighbourhood T_{iter}.
- How is it fixed? Experimentally...

First questionning

- The VPLS matheuristic requests fixing several parameters:
  1. The window size h,
  2. The total allowed CPU time T_{stop},
  3. The allowed CPU time for exploring a neighbourhood T_{iter}.
- How is it fixed? Experimentally...
  1. $h = 12$,
  2. $T_{stop} = 60, 600, 3600s$ (depending on n)
  3. $T_{iter} = 10, 60, 100s$ (depending on n).

First questionning

- The VPLS matheuristic requests fixing several parameters:
  1. The window size h,
  2. The total allowed CPU time T_{stop},
  3. The allowed CPU time for exploring a neighbourhood T_{iter}.
- How is it fixed? Experimentally...
  1. $h = 12$,
  2. $T_{stop} = 60, 600, 3600s$ (depending on n)
  3. $T_{iter} = 10, 60, 100s$ (depending on n).
- "Preliminary experiments"... a classic trick to mean "rule of thumb".

First questionning

- The VPLS matheuristic requests fixing several parameters:
  1. The window size h,
  2. The total allowed CPU time T_{stop},
  3. The allowed CPU time for exploring a neighbourhood T_{iter}.
- How is it fixed? Experimentally...
  1. $h = 12$,
  2. $T_{stop} = 60, 600, 3600s$ (depending on n)
  3. $T_{iter} = 10, 60, 100s$ (depending on n).
- "Preliminary experiments"... a classic trick to mean "rule of thumb".
- General question: how to fix the parameters of a heuristic?

First questionning

- The VPLS matheuristic requests fixing several parameters:
  1. The window size h,
  2. The total allowed CPU time T_{stop},
  3. The allowed CPU time for exploring a neighbourhood T_{iter}.
- How is it fixed? Experimentally...
  1. $h = 12$,
  2. $T_{stop} = 60, 600, 3600s$ (depending on n)
  3. $T_{iter} = 10, 60, 100s$ (depending on n).
- "Preliminary experiments"... a classic trick to mean "rule of thumb".
- General question: how to fix the parameters of a heuristic?
- We answer this question by means of the bayesian optimization.

Principles of bayesian optimization

The goal is to determine where is the maximum of a continuous function f:

$$x^+ = arg(\max_{x \in A}(f(x))$$

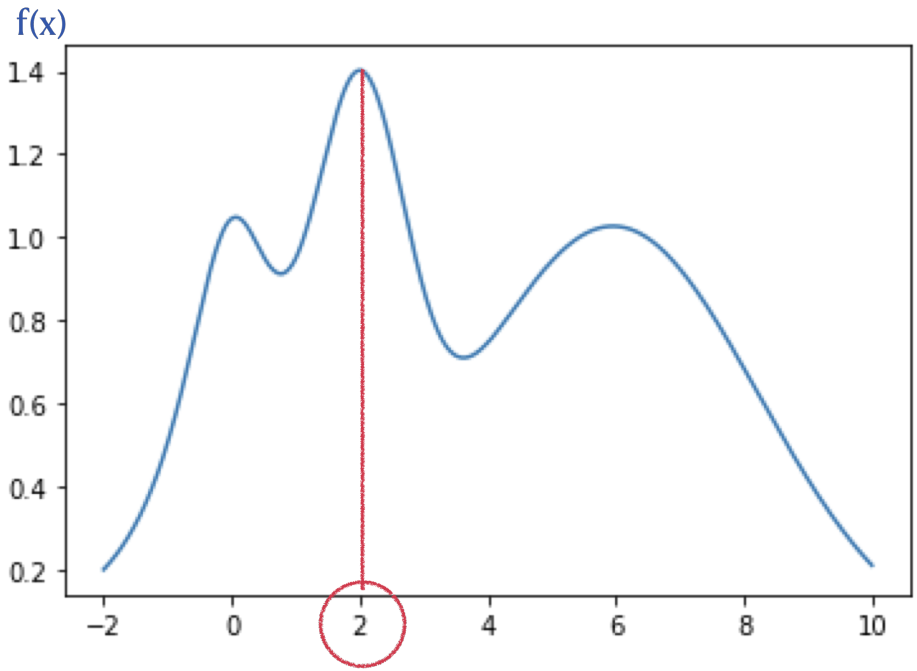


Figure: In that example, max (f(x)) is at x=2, A=[-2;10]

Principles of bayesian optimization

The goal is to determine where is the maximum of a continuous function f:

$$x^+ = arg(\max_{x \in A}(f(x))$$

- We guess the maximum by sampling the function (while minimizing the number of guesses).

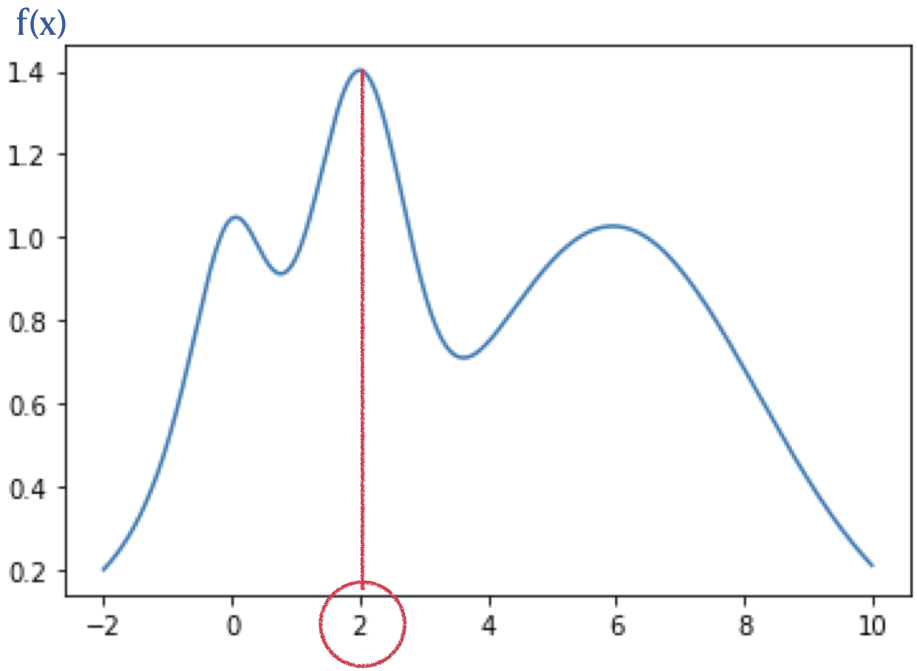


Figure: In that example, max (f(x)) is at x=2, A=[-2;10]

Principles of bayesian optimization

The goal is to determine where is the maximum of a continuous function f:

$$x^+ = arg(\max_{x \in A}(f(x))$$

- We guess the maximum by sampling the function (while minimizing the number of guesses).
- It is adapted to multi-variable functions.

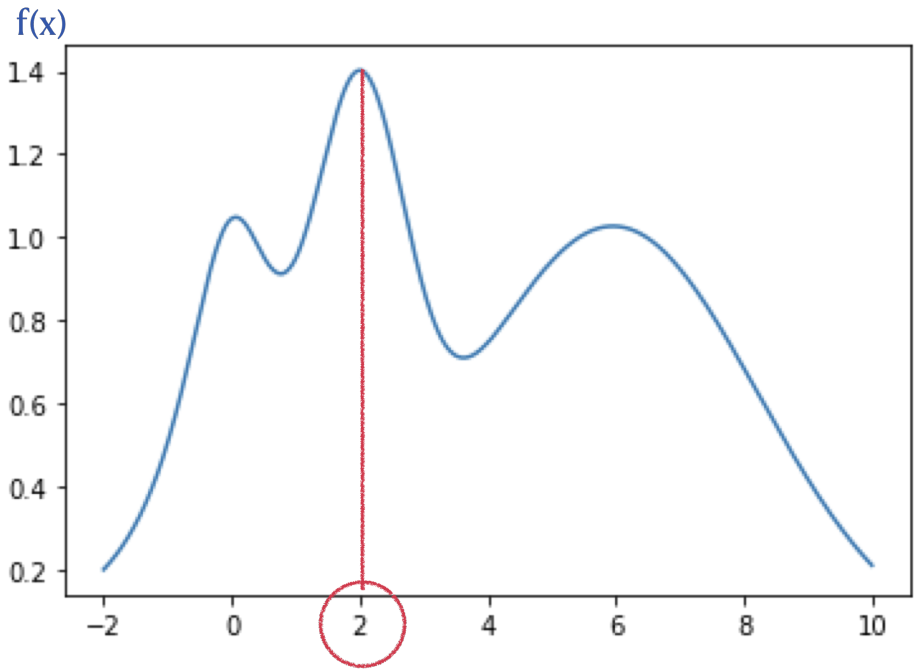


Figure: In that example, max (f(x)) is at x=2, A=[-2;10]

Principles of bayesian optimization

- An iterative process,

[13] Brochu, E., Cora, V.M., and de Freitas, N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. pre-print, 2010. arXiv:1012.2599

Principles of bayesian optimization

- An iterative process,
- At each iteration t, we know $D_{1:t}$ the sample observations of $f(x)$ computed so far,

[13] Brochu, E., Cora, V.M., and de Freitas, N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. pre-print, 2010. arXiv:1012.2599

Principles of bayesian optimization

- An iterative process,
- At each iteration t, we know $D_{1:t}$ the sample observations of $f(x)$ computed so far,
- We make the assumption that function f follows a Gaussian process (GP, [13]):

$$f(x) \approx GP(\mu(x), k(x, x'))$$

with μ the mean function and k the covariance function used to model GP.

[13] Brochu, E., Cora, V.M., and de Freitas, N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. pre-print, 2010. arXiv:1012.2599

Principles of bayesian optimization

- An iterative process,
- At each iteration t, we know $D_{1:t}$ the sample observations of $f(x)$ computed so far,
- We make the assumption that function f follows a Gaussian process (GP, [13]):

$$f(x) \approx GP(\mu(x), k(x, x'))$$

with μ the mean function and k the covariance function used to model GP.

$\Rightarrow$ GP returns the expected value $\mu(x)$ of $f(x)$ and the associated variance $\sigma(x)$

[13] Brochu, E., Cora, V.M., and de Freitas, N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. pre-print, 2010. arXiv:1012.2599

Principles of bayesian optimization

- $\forall x_{t'} \in D_{1:t}$, $\mu(x_{t'}) = f(x_{t'})$ and $\sigma(x_{t'}) = 0$,

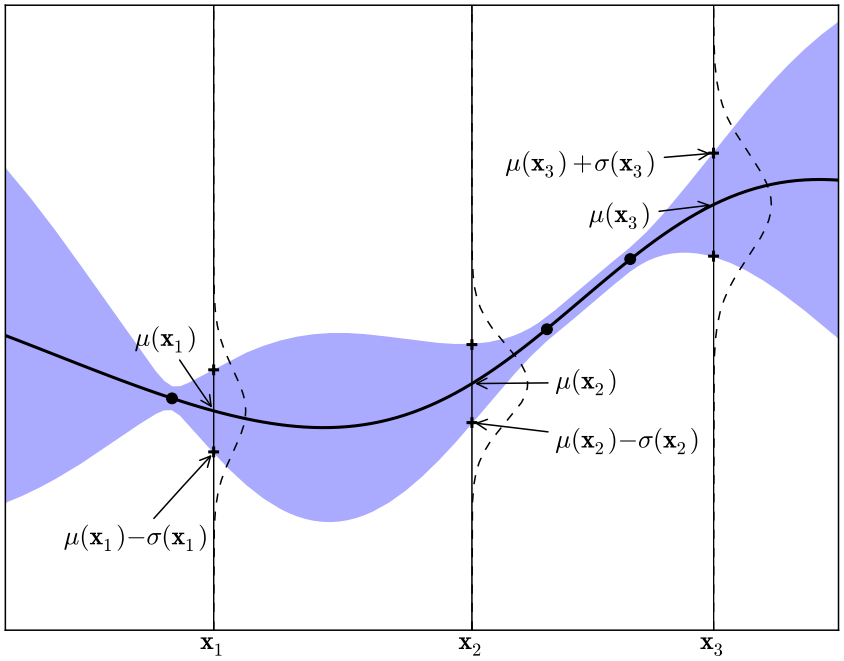


Figure: mean (black), variance (blue)

Principles of bayesian optimization

- $\forall x_{t'} \in D_{1:t}$, $\mu(x_{t'}) = f(x_{t'})$ and $\sigma(x_{t'}) = 0$,
- From set $D_{1:t}$ it is possible to evaluate the mean (black curve) and the variance (blue area)... so to evaluate the Gaussian process.

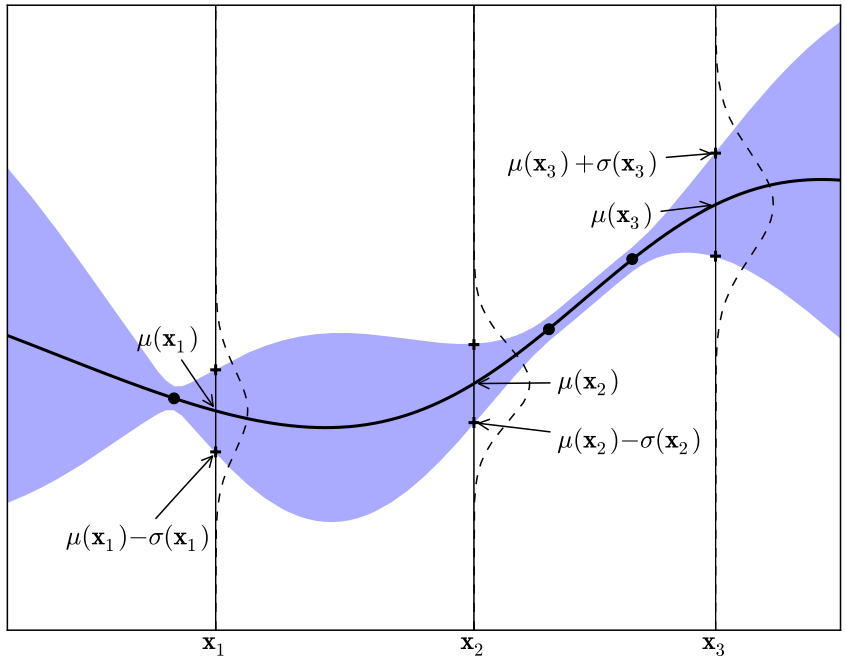
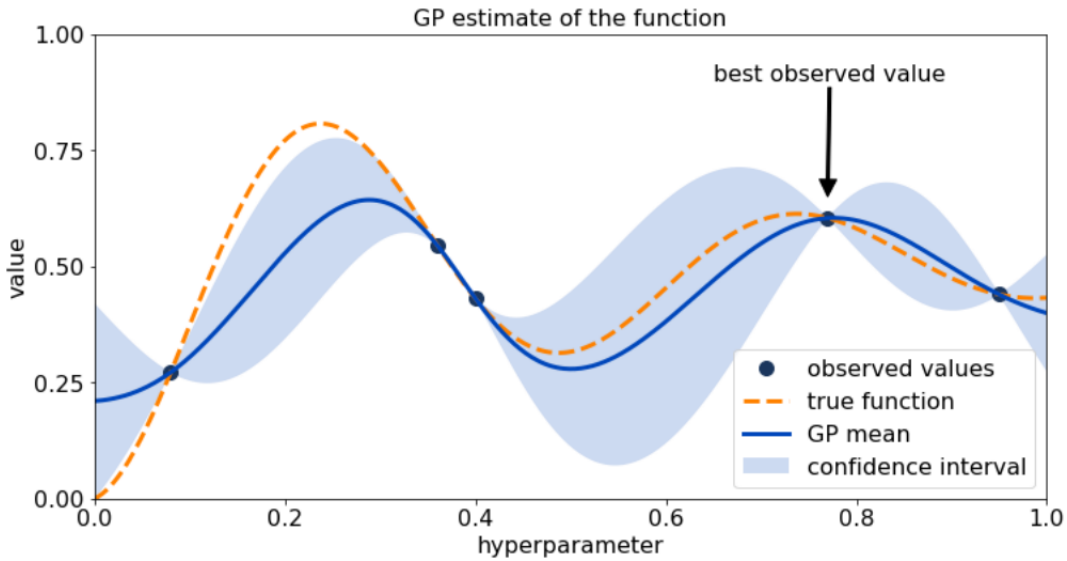


Figure: mean (black), variance (blue)

Principles of bayesian optimization

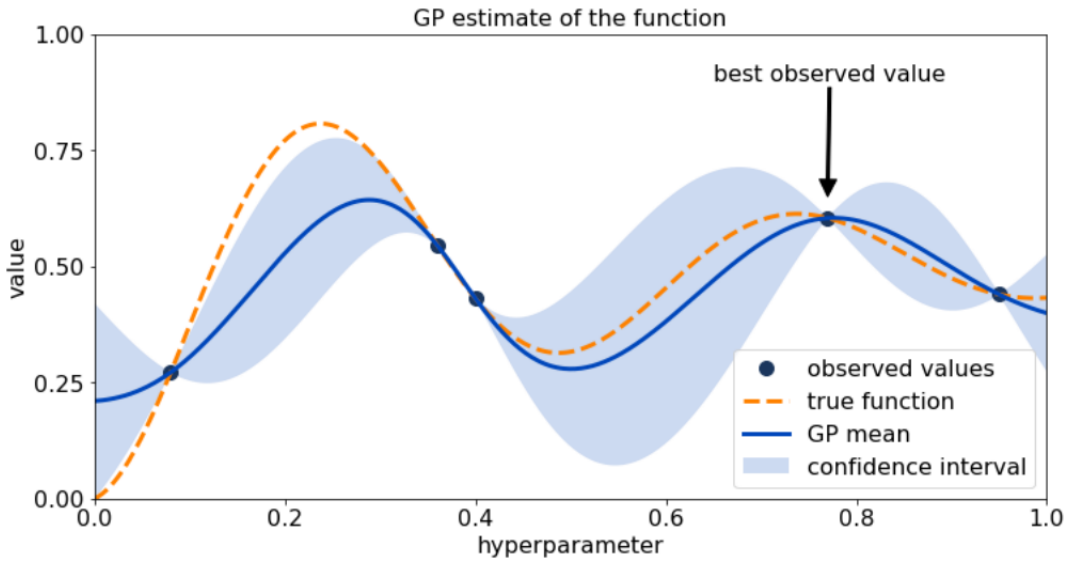
- How to choose the next point x_{t+1} for which we compute f?



https://cds.cern.ch/record/2702355/plots

Principles of bayesian optimization

- How to choose the next point x_{t+1} for which we compute f?

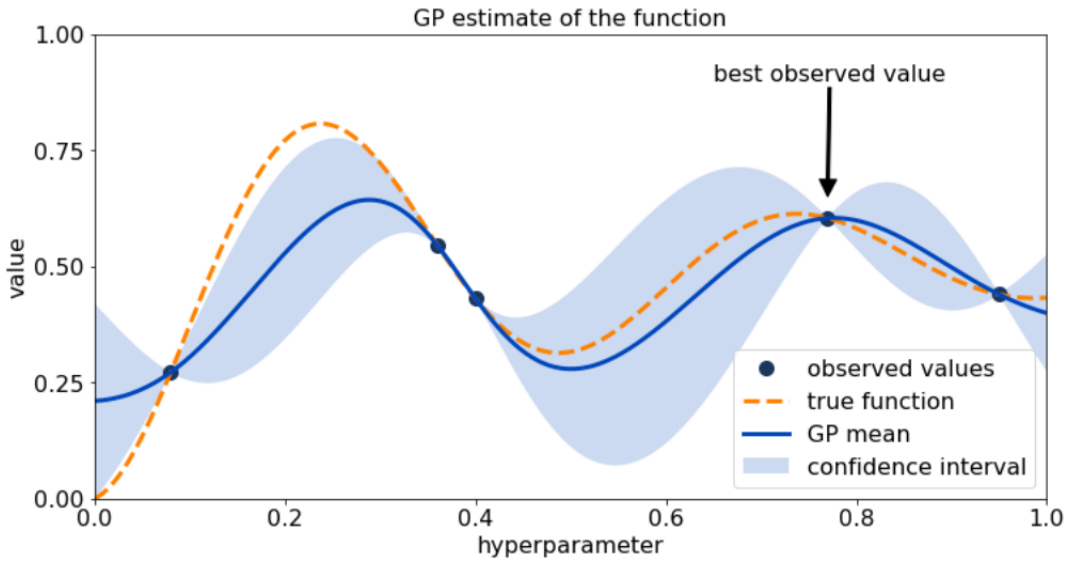


https://cds.cern.ch/record/2702355/plots

- Two strategies: intensification (try where you think the optimal is) or diversification (try where the variance is maximum).

Principles of bayesian optimization

- How to choose the next point x_{t+1} for which we compute f?



https://cds.cern.ch/record/2702355/plots

- Two strategies: intensification (try where you think the optimal is) or diversification (try where the variance is maximum).
- The choice of x is lead by the *acquisition function*... different versions exist (PI, EI, UCB, ...)

Principles of bayesian optimization

- **GP-UCB** : Upper Confidence Bound : determines whether the next sampling point should make use of the current optimum value (corresponding to the high $\mu(x)$ zone) or should explore other low confidence zone (corresponding to the high $\sigma(x)$ zone). The parameter k is used to make a trade-off between them:

$$UCB(x) = \mu(x) + k.\sigma(x)$$

- Other acquisition functions exist.

Principles of bayesian optimization: the algorithm

1. Set $t = 1$, $D_{1:t} = \{(x_0, f(x_0); (x_1, f(x_1))\}$ with x_0 and x_1 picked at random,

Principles of bayesian optimization: the algorithm

1. Set $t = 1$, $D_{1:t} = \{(x_0, f(x_0); (x_1, f(x_1))\}$ with x_0 and x_1 picked at random,
2. While (not the end)
   1. $t = t + 1$,
   2. Use the acquisition function to generate x_t,
   3. $D_{1:t} = D_{1:(t-1)} + \{(x_t, f(x_t))\}$,
3. Return $x = argmax_{x' \in D_{1:t}}(f(x'))$

Application to MATH: the parameters

- Parameters to learn: h, T_{stop} and T_{iter}

Application to MATH: the parameters

- Parameters to learn: h, T_{stop} and T_{iter}
- We make the following assumptions:

Application to MATH: the parameters

- Parameters to learn: h, T_{stop} and T_{iter}
- We make the following assumptions:
  1. h: can't be too large $\Rightarrow$ independent from n,

Application to MATH: the parameters

- Parameters to learn: h, T_{stop} and T_{iter}
- We make the following assumptions:
  1. h: can't be too large $\Rightarrow$ independent from n,
  2. T_{iter}: depends on the value of h,

Application to MATH: the parameters

- Parameters to learn: h, T_{stop} and T_{iter}
- We make the following assumptions:
  1. h: can't be too large $\Rightarrow$ independent from n,
  2. T_{iter}: depends on the value of h,
  3. T_{stop}: depends on the values of n and T_{iter}.

Application to MATH: the parameters

- Parameters to learn: h, T_{stop} and T_{iter}
- We make the following assumptions:
  1. h: can't be too large $\Rightarrow$ independent from n,
  2. T_{iter}: depends on the value of h,
  3. T_{stop}: depends on the values of n and T_{iter}.
- We define:

Application to MATH: the parameters

- Parameters to learn: h, T_{stop} and T_{iter}
- We make the following assumptions:
  1. h: can't be too large $\Rightarrow$ independent from n,
  2. T_{iter}: depends on the value of h,
  3. T_{stop}: depends on the values of n and T_{iter}.
- We define:
  1. $T_{iter} = \beta_1 h + \beta_2$,

Application to MATH: the parameters

- Parameters to learn: h, T_{stop} and T_{iter}
- We make the following assumptions:
  1. h: can't be too large $\Rightarrow$ independent from n,
  2. T_{iter}: depends on the value of h,
  3. T_{stop}: depends on the values of n and T_{iter}.
- We define:
  1. $T_{iter} = \beta_1 h + \beta_2$,
  2. $T_{stop} = \gamma_1 n + \gamma_2 T_{iter} + \gamma_3$.

Application to MATH: the parameters

- Parameters to learn: h, T_{stop} and T_{iter}
- We make the following assumptions:
  1. h: can't be too large $\Rightarrow$ independent from n,
  2. T_{iter}: depends on the value of h,
  3. T_{stop}: depends on the values of n and T_{iter}.
- We define:
  1. $T_{iter} = \beta_1 h + \beta_2$,
  2. $T_{stop} = \gamma_1 n + \gamma_2 T_{iter} + \gamma_3$.

$\Rightarrow$ We want to learn $h, \beta_1, \beta_2, \gamma_1, \gamma_2$ and γ_3.

Application to MATH: the parameters

- Parameters to learn: h, T_{stop} and T_{iter}
- We make the following assumptions:
  1. h: can't be too large $\Rightarrow$ independent from n,
  2. T_{iter}: depends on the value of h,
  3. T_{stop}: depends on the values of n and T_{iter}.
- We define:
  1. $T_{iter} = \beta_1 h + \beta_2$,
  2. $T_{stop} = \gamma_1 n + \gamma_2 T_{iter} + \gamma_3$.

  $\Rightarrow$ We want to learn $h, \beta_1, \beta_2, \gamma_1, \gamma_2$ and γ_3.
- We learn $x^* = [h, \beta_1, \beta_2, \gamma_1, \gamma_2, \gamma_3]$ by Bayesian optimization: what is the function f to maximize?

Application to MATH: the function

- We first need to introduce computational elements,

[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research*, 213:67-78.

Application to MATH: the function

- We first need to introduce computational elements,
- We generate a database for learning:

[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research*, 213:67-78.

Application to MATH: the function

- We first need to introduce computational elements,
- We generate a database for learning:
  - $\mathcal{B}_{learn}$: 300 instances with 50 jobs plus 300 instances with 100 jobs,

[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research*, 213:67-78.

Application to MATH: the function

- We first need to introduce computational elements,
- We generate a database for learning:
  - $\mathcal{B}_{learn}$: 300 instances with 50 jobs plus 300 instances with 100 jobs,
  - Each instance is generated following the classic protocol described in [2]: $p_{i,j} \in U[1;100]$,

[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research*, 213:67-78.

Application to MATH: the function

- We first need to introduce computational elements,
- We generate a database for learning:
  - $\mathcal{B}_{learn}$: 300 instances with 50 jobs plus 300 instances with 100 jobs,
  - Each instance is generated following the classic protocol described in [2]: $p_{i,j} \in U[1;100]$,
  - To each instance, we want to associate the value $\sum_j C_j^M$ of MATH.

[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research*, 213:67-78.

Application to MATH: the function

- We first need to introduce computational elements,
- We generate a database for learning:
  - $\mathcal{B}_{learn}$: 300 instances with 50 jobs plus 300 instances with 100 jobs,
  - Each instance is generated following the classic protocol described in [2]: $p_{i,j} \in U[1;100]$,
  - To each instance, we want to associate the value $\sum_j C_j^M$ of MATH.
- How to cope with the randomness inside the matheuristics?

[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research*, 213:67-78.

Application to MATH: the function

- We first need to introduce computational elements,
- We generate a database for learning:
  - $\mathcal{B}_{learn}$: 300 instances with 50 jobs plus 300 instances with 100 jobs,
  - Each instance is generated following the classic protocol described in [2]: $p_{i,j} \in U[1;100]$,
  - To each instance, we want to associate the value $\sum_j C_j^M$ of MATH.
- How to cope with the randomness inside the matheuristics?

  On each instance, 3 runs of a matheuristic $\Rightarrow$ we average the returned objective function values.

[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research*, 213:67-78.

Application to MATH: the function

- We first need to introduce computational elements,
- We generate a database for learning:
  - $\mathcal{B}_{learn}$: 300 instances with 50 jobs plus 300 instances with 100 jobs,
  - Each instance is generated following the classic protocol described in [2]: $p_{i,j} \in U[1;100]$,
  - To each instance, we want to associate the value $\sum_j C_j^M$ of MATH.
- How to cope with the randomness inside the matheuristics?

  On each instance, 3 runs of a matheuristic $\Rightarrow$ we average the returned objective function values.
- The averaged $\sum_j C_j$'s of MATH are precomputed and stored in $\mathcal{B}_{learn}$.

[2] Della Croce, F., Grosso, A., Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research*, 213:67-78.

Application to MATH: the function

Parameter tuning optimization problem,

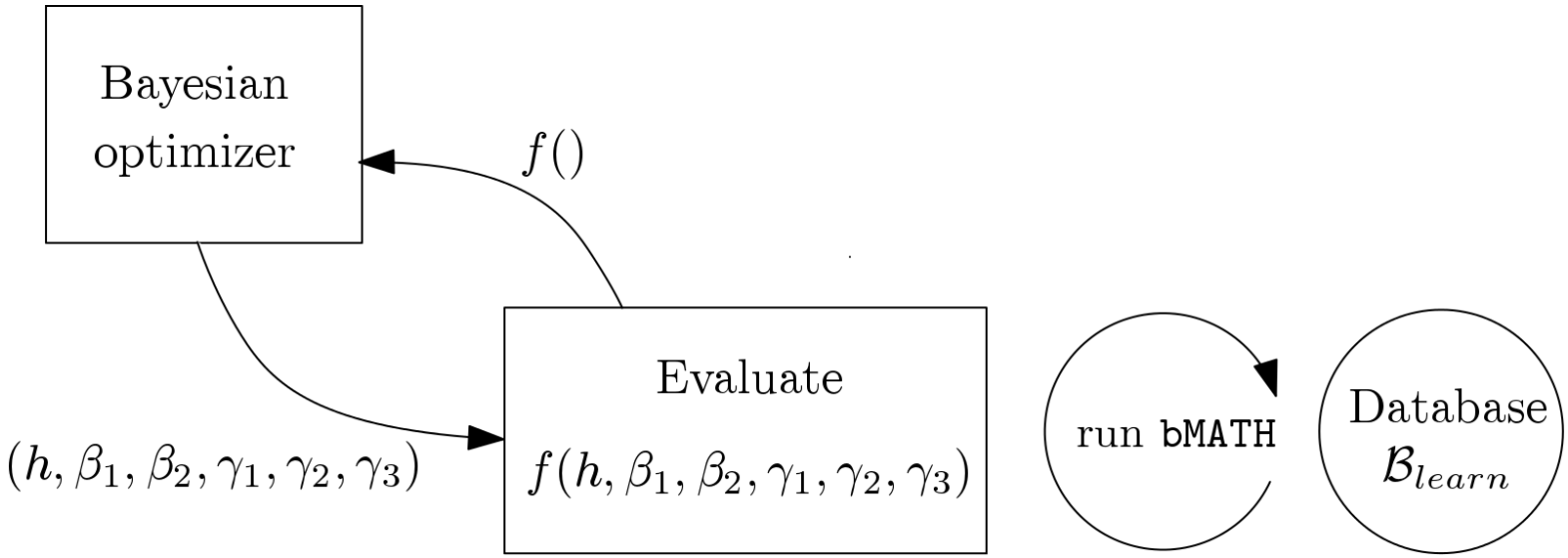
Maximize $f(h, \beta_1, \beta_2, \gamma_1, \gamma_2, \gamma_3) = \sum_{I \in \mathcal{B}_{learn}} \frac{\sum_j C_j^M - \sum_j C_j^{bM}}{\sum_j C_j^M}$
$-0.005\, T_{iter} - 0.001\, T_{stop}$

st

$h, \beta_1, \beta_2, \gamma_1, \gamma_2, \gamma_3 \in \mathbb{R}$

with $\sum_j C_j^M$ (resp. $\sum_j C_j^{bM}$) the value of $\sum_j C_j$ computed by the original (resp. bayesian with parameters $(h, \beta_1, \beta_2, \gamma_1, \gamma_2, \gamma_3)$) matheuristic on instance I,

Application to MATH: the function



Tuning the Bayesian optimizer

The Bayesian optimizer is tuned as follows:

- Python library "Bayesian Optimization"
- Acquisition function: GP-UCB, with $\kappa = 2.576$
- $6 < h < 18$
- $0 < \beta_1 < 0.2$
- $0.1 < \beta_2 < 1$
- $0 < \gamma_1 < 1$
- $0 < \gamma_2 < 1$
- $10 < \gamma_3 < 300$

Experimental results

- The set of computed parameters:

Experimental results

- The set of computed parameters:
  1. $h = 14.405 \Rightarrow h = 14$,

Experimental results

- The set of computed parameters:
  1. $h = 14.405 \Rightarrow h = 14$,
  2. $\beta_1 = 0.0718$, $\beta_2 = 0.3057 \Rightarrow T_{iter} = 1.3s$,

Experimental results

- The set of computed parameters:
  1. $h = 14.405 \Rightarrow h = 14$,
  2. $\beta_1 = 0.0718$, $\beta_2 = 0.3057 \Rightarrow T_{iter} = 1.3s$,
  3. $\gamma_1 = 0.3217$, $\gamma_2 = 0.4377$, $\gamma_3 = 11.0903 \Rightarrow T_{stop} = 0.44n + 11.5s$.

Experimental results

- The set of computed parameters:
  1. $h = 14.405 \Rightarrow h = 14$,
  2. $\beta_1 = 0.0718$, $\beta_2 = 0.3057 \Rightarrow T_{iter} = 1.3s$,
  3. $\gamma_1 = 0.3217$, $\gamma_2 = 0.4377$, $\gamma_3 = 11.0903 \Rightarrow T_{stop} = 0.44n + 11.5s$.
- Comparison table:

n	MATH			bMATH		
	h	T_{iter}	T_{stop}	h	T_{iter}	T_{stop}
100	12	10	60	14	1.3	55.5
300	12	60	600	14	1.3	143.5
500	12	100	3600	14	1.3	231.5

Experimental results

- Statistics on the Bayesian optimization process:

Experimental results

- Statistics on the Bayesian optimization process:
  - 25 iterations done,

Experimental results

- Statistics on the Bayesian optimization process:
  - 25 iterations done,
  - An iteration: between 2h (rare) and 1 day (often),

Experimental results

- Statistics on the Bayesian optimization process:
  - 25 iterations done,
  - An iteration: between 2h (rare) and 1 day (often),
  - Total CPU time $\approx$ 20 days,

Experimental results

- Statistics on the Bayesian optimization process:
  - 25 iterations done,
  - An iteration: between 2h (rare) and 1 day (often),
  - Total CPU time $\approx$ 20 days,
  - By chance (?), the best parameters were found at iteration...2!

Experimental results

- We now compare three versions of the matheuristic:

Experimental results

- We now compare three versions of the matheuristic:
  1. The original `MATH` heuristic,

Experimental results

- We now compare three versions of the matheuristic:
  1. The original MATH heuristic,
  2. $\texttt{MATH}_{14}$: a version of MATH in which we set $h = 14$,

Experimental results

- We now compare three versions of the matheuristic:
  1. The original MATH heuristic,
  2. $\texttt{MATH}_{14}$: a version of MATH in which we set $h = 14$,
  3. bMATH: the matheuristic with the learned parameters.

Experimental results

- We now compare three versions of the matheuristic:
  1. The original MATH heuristic,
  2. $\texttt{MATH}_{14}$: a version of MATH in which we set $h = 14$,
  3. bMATH: the matheuristic with the learned parameters.
- For each problem size, 50 instances are randomly generated,

n	MATH			bMATH		
	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)
50	61.17	0.0058	0.0689	34.35	0.0030	0.0350
100	61.28	0.0047	0.0290	56.39	0.0040	0.0140
150	646.62	0.0387	1.6107	128.951	0.0144	0.0481

Deviation $\delta = \frac{H - BestUB}{BestUB}$, with $BestUB = \min(\sum_j C_j^M, \sum_j C_j^{M_{14}}, \sum_j C_j^{bM})$.

Experimental results

- We now compare three versions of the matheuristic:
  1. The original MATH heuristic,
  2. $\texttt{MATH}_{14}$: a version of MATH in which we set $h = 14$,
  3. bMATH: the matheuristic with the learned parameters.
- For each problem size, 50 instances are randomly generated,

n	MATH			bMATH		
	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)
50	61.17	0.0058	0.0689	34.35	0.0030	0.0350
100	61.28	0.0047	0.0290	56.39	0.0040	0.0140
150	646.62	0.0387	1.6107	128.951	0.0144	0.0481

Deviation $\delta = \frac{H - BestUB}{BestUB}$, with $BestUB = \min(\sum_j C_j^M, \sum_j C_j^{M_{14}}, \sum_j C_j^{bM})$.

Experimental results

- We now compare three versions of the matheuristic:
  1. The original MATH heuristic,
  2. $\texttt{MATH}_{14}$: a version of MATH in which we set $h = 14$,
  3. bMATH: the matheuristic with the learned parameters.
- For each problem size, 50 instances are randomly generated,

n	MATH			bMATH		
	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)
50	61.17	0.0058	0.0689	34.35	0.0030	0.0350
100	61.28	0.0047	0.0290	56.39	0.0040	0.0140
150	646.62	0.0387	1.6107	128.951	0.0144	0.0481

Deviation $\delta = \frac{H - BestUB}{BestUB}$, with $BestUB = \min(\sum_j C_j^M, \sum_j C_j^{M_{14}}, \sum_j C_j^{bM})$.

n	$\texttt{MATH}_{14}$		
	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)
50	61.39	0.0013	0.0190
100	61.50	0.0038	0.0249
150	601.91	0.0015	0.0215

- Does not scale for larger n: computation of T_{iter} must also depend on n.

Conclusions about Bayesian optimization

- Bayesian optimization seems to be a good tool to determine suitable parameters,

Conclusions about Bayesian optimization

- Bayesian optimization seems to be a good tool to determine suitable parameters,
- Learning the parameters can be a bit long (days/weeks),

Conclusions about Bayesian optimization

- Bayesian optimization seems to be a good tool to determine suitable parameters,
- Learning the parameters can be a bit long (days/weeks),
- Technical conclusions:

Conclusions about Bayesian optimization

- Bayesian optimization seems to be a good tool to determine suitable parameters,
- Learning the parameters can be a bit long (days/weeks),
- Technical conclusions:
  1. The database for learning must be representative: maybe not always easy to generate,

Conclusions about Bayesian optimization

- Bayesian optimization seems to be a good tool to determine suitable parameters,
- Learning the parameters can be a bit long (days/weeks),
- Technical conclusions:
  1. The database for learning must be representative: maybe not always easy to generate,
  2. Requires an appropriate modeling of the parameters to learn.

Outline

A light in the dark

- A matheuristic, as any local search, blindly explores neighbourhoods,

A light in the dark

- A matheuristic, as any local search, blindly explores neighbourhoods,
- A central questionning: how to choose the next neightbourhood to explore?

A light in the dark

- A matheuristic, as any local search, blindly explores neighbourhoods,
- A central questionning: how to choose the next neightbourhood to explore?
- Trust your chance: pick neighbourhoods at random!

A light in the dark

- A matheuristic, as any local search, blindly explores neighbourhoods,
- A central questionning: how to choose the next neightbourhood to explore?
- Trust your chance: pick neighbourhoods at random!
- What we would like, ideally, is to find out the best neighbourhood (window of positions),

A light in the dark

- A matheuristic, as any local search, blindly explores neighbourhoods,
- A central questionning: how to choose the next neightbourhood to explore?
- Trust your chance: pick neighbourhoods at random!
- What we would like, ideally, is to find out the best neighbourhood (window of positions),
- Ideal goal: to have an oracle (predictor) capable of predicting, for a given s, the best values of r and h (neighbourhood containing the best solution),

A light in the dark

- A matheuristic, as any local search, blindly explores neighbourhoods,
- A central questionning: how to choose the next neightbourhood to explore?
- Trust your chance: pick neighbourhoods at random!
- What we would like, ideally, is to find out the best neighbourhood (window of positions),
- Ideal goal: to have an oracle (predictor) capable of predicting, for a given s, the best values of r and h (neighbourhood containing the best solution),
- Reasonable goal: design, for given r, h and s, an oracle predicting how much the best solution of this neighbourhood improves upon s,

A light in the dark

- Use of structured machine learning to solve this *regression* problem,

A light in the dark

- Use of structured machine learning to solve this *regression* problem,
- We try to build an oracle capable of predicting it,

A light in the dark

- Use of structured machine learning to solve this *regression* problem,
- We try to build an oracle capable of predicting it,
- Two Machine Learning approaches:

A light in the dark

- Use of structured machine learning to solve this *regression* problem,
- We try to build an oracle capable of predicting it,
- Two Machine Learning approaches:
  1. A classic features-based approach,

A light in the dark

- Use of structured machine learning to solve this *regression* problem,
- We try to build an oracle capable of predicting it,
- Two Machine Learning approaches:
  1. A classic features-based approach,
  2. A neural network approach that exploits the sequence of jobs only.

A feature-based neural network approach

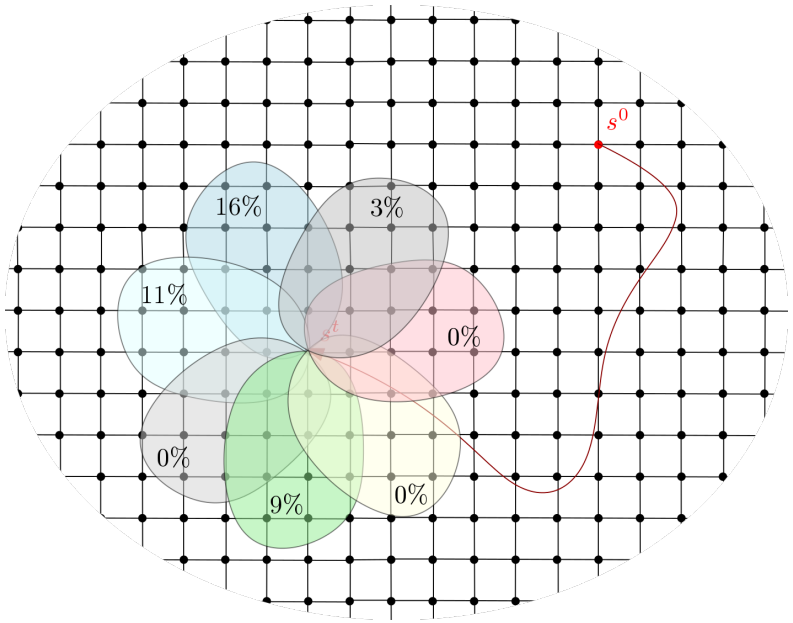
- To any $x = [r; h; s]$, we associate a vector $\phi(x) \in \mathbb{R}^{87}$ of 87 features,

A feature-based neural network approach

- To any $x = [r; h; s]$, we associate a vector $\phi(x) \in \mathbb{R}^{87}$ of 87 features,
- We want to build a predictor $p(\phi(x), \theta^*) \in [0; 100]$, with $\theta^* \in \Theta$.

A feature-based neural network approach

- To any $x = [r; h; s]$, we associate a vector $\phi(x) \in \mathbb{R}^{87}$ of 87 features,
- We want to build a predictor $p(\phi(x), \theta^*) \in [0; 100]$, with $\theta^* \in \Theta$.
  - $\Rightarrow$ $p(\phi(x), \theta^*)$ corresponds to the percentage of improvement over s of the best solution in the neighbourhood defined by r and h.



A feature-based neural network approach

- Predictor $p(\phi(x), \theta)$ is a neural network and the θ weights (Deep Learning),

A feature-based neural network approach

- Predictor $p(\phi(x), \theta)$ is a neural network and the θ weights (Deep Learning),
- The main point: how to learn θ^* ?

A feature-based neural network approach

- The main point: how to learn θ^* ?

A feature-based neural network approach

- The main point: how to learn θ^* ?
- First, generate:
  - A training database $\mathcal{D}_{train} = \{\phi(x_m), y_m\}_{m=1}^{M_1}$,

with y_m is the percentage of improvement over s_m of the best solution obtained after reoptimizing in $[r_m; r_m + h_m]$.

A feature-based neural network approach

- The main point: how to learn θ^* ?
- First, generate:
  - A training database $\mathcal{D}_{train} = \{\phi(x_m), y_m\}_{m=1}^{M_1}$,
  - A validation database $\mathcal{D}_{valid} = \{\phi(x_m), y_m\}_{m=1}^{M_2}$,

with y_m is the percentage of improvement over s_m of the best solution obtained after reoptimizing in $[r_m; r_m + h_m]$.

A feature-based neural network approach

- The main point: how to learn θ^* ?
- First, generate:
  - A training database $\mathcal{D}_{train} = \{\phi(x_m), y_m\}_{m=1}^{M_1}$,
  - A validation database $\mathcal{D}_{valid} = \{\phi(x_m), y_m\}_{m=1}^{M_2}$,
  - A test database $\mathcal{D}_{test} = \{\phi(x_m), y_m\}_{m=1}^{M_3}$,

  with y_m is the percentage of improvement over s_m of the best solution obtained after reoptimizing in $[r_m; r_m + h_m]$.

A feature-based neural network approach

- The main point: how to learn θ^* ?
- First, generate:
  - A training database $\mathcal{D}_{train} = \{\phi(x_m), y_m\}_{m=1}^{M_1}$,
  - A validation database $\mathcal{D}_{valid} = \{\phi(x_m), y_m\}_{m=1}^{M_2}$,
  - A test database $\mathcal{D}_{test} = \{\phi(x_m), y_m\}_{m=1}^{M_3}$,

  with y_m is the percentage of improvement over s_m of the best solution obtained after reoptimizing in $[r_m; r_m + h_m]$.
- Training of the model is done on $\mathcal{D}_{train}$ while computing:

$$\min_{\theta \in \Theta} \sum_{(\phi(x_m), y_m) \in \mathcal{D}_{train}} \mathcal{L}(p(\phi(x_m), \theta), y_m)$$

  $\mathcal{L}$ being a loss function gauging the error between the predicted value and the real value (y_m).

A feature-based neural network approach

- A standard loss function for regression problems was chosen: Mean Square Error,

$$\mathcal{L}(x, y) = (x - y)^2.$$

A feature-based neural network approach

- A standard loss function for regression problems was chosen: Mean Square Error,

$$\mathcal{L}(x, y) = (x - y)^2.$$

- This optimization problem is solved by an iterative gradient descent method called Adam.

A feature-based neural network approach

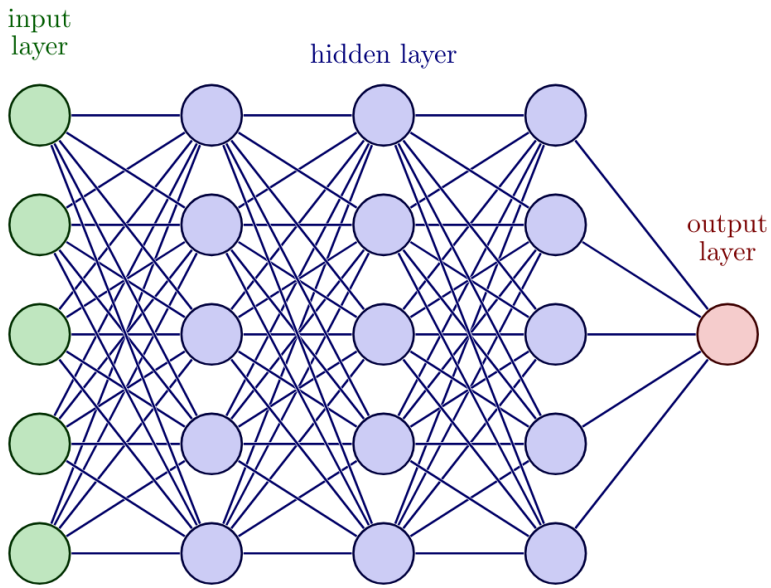
- A standard loss function for regression problems was chosen: Mean Square Error,

$$\mathcal{L}(x, y) = (x - y)^2.$$

- This optimization problem is solved by an iterative gradient descent method called Adam.
- We keep the set of weights θ^* that **minimizes the valueo of the loss function on** $\mathcal{D}_{valid}$.

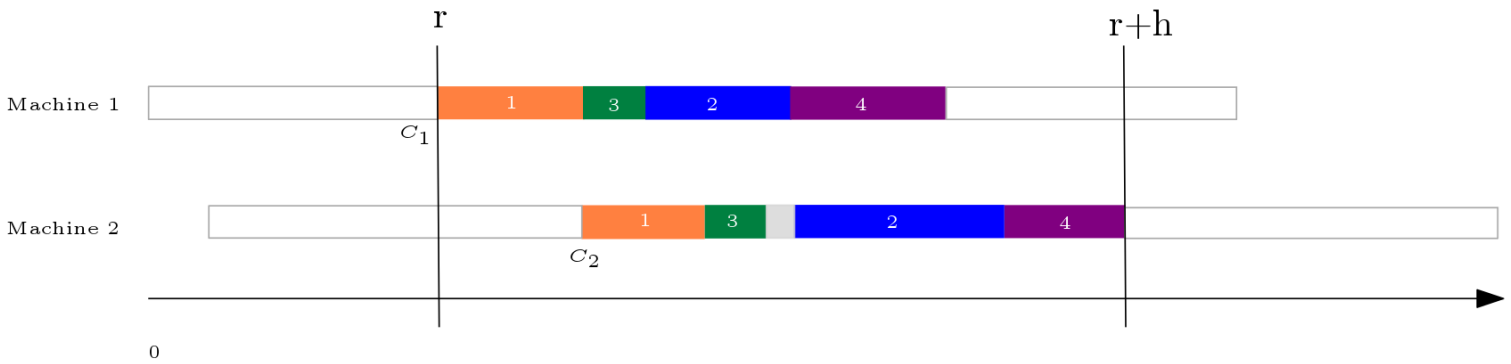
A feature-based neural network approach

- Predictor (p) is a fully connected neural network with 5 layers with 140 neurons each and an output layer (determined by preliminary experiments),
- Overfitting breakers : Batchnorm, L1 regularization,
- Dimension of the weights vector θ: 94781,



A feature-based neural network approach

- A set of 87 features,



Descriptive features:

- C_1, C_2, $\sum_{j=r}^{r+h} p_{s[j],1}$, $\sum_{j=r}^{r+h} p_{s[j],2}$,
- In $[r; r+h]$: ratios $\frac{p_{j,1}}{p_{j,2}}$, idle times on M_2, number of jobs not in SPT order on M_2, ...
- In $[r+h+1; n]$: idle times on M_2.

Informative features:

- Upper bound on the gain (on $\sum_{j=r+h+1}^{n} C_j$) in rescheduling $[r; r+h]$,
- Lower bounds on the gain (on $\sum_{j=r}^{r+h} C_j$) in rescheduling $[r; r+h]$,
- Upper bounds on the gain (on $\sum_{j=r}^{r+h} C_j$) in rescheduling $[r; r+h]$,

- Features are normalized and standardized.

Building the predictor: the databases

- To generate the *training*, *validation* and *test* databases, the same protocol has been used,

Building the predictor: the databases

- To generate the *training*, *validation* and *test* databases, the same protocol has been used,
- For each database,
  1. Randomly generate 4000 instances of the scheduling problem for a given $n = 60$ and 2000 instances with $n = 100$,

Building the predictor: the databases

- To generate the *training*, *validation* and *test* databases, the same protocol has been used,
- For each database,
  1. Randomly generate 4000 instances of the scheduling problem for a given $n = 60$ and 2000 instances with $n = 100$,
  2. Run `MATH` in which all windows $[r; r + 14]$ are tested. For each $x = [r; 14; s]$ record $\phi(x)$ and y the percentage of improvement achieved by optimizing this window ($h = 14$ comes from Bayesian optimization),

Building the predictor: the databases

- To generate the *training*, *validation* and *test* databases, the same protocol has been used,
- For each database,
  1. Randomly generate 4000 instances of the scheduling problem for a given $n = 60$ and 2000 instances with $n = 100$,
  2. Run `MATH` in which all windows $[r; r + 14]$ are tested. For each $x = [r; 14; s]$ record $\phi(x)$ and y the percentage of improvement achieved by optimizing this window ($h = 14$ comes from Bayesian optimization),

Note: time to generate a single database $\approx$ 1-3 month.

Building the predictor: the databases

- To generate the *training*, *validation* and *test* databases, the same protocol has been used,
- For each database,
  1. Randomly generate 4000 instances of the scheduling problem for a given $n = 60$ and 2000 instances with $n = 100$,
  2. Run `MATH` in which all windows $[r; r + 14]$ are tested. For each $x = [r; 14; s]$ record $\phi(x)$ and y the percentage of improvement achieved by optimizing this window ($h = 14$ comes from Bayesian optimization),

  Note: time to generate a single database $\approx$ 1-3 month.
- Sizes of the databases $\approx$ 3M schedules/windows,

Building the predictor: the databases

- To generate the *training*, *validation* and *test* databases, the same protocol has been used,
- For each database,
  1. Randomly generate 4000 instances of the scheduling problem for a given $n = 60$ and 2000 instances with $n = 100$,
  2. Run `MATH` in which all windows $[r; r + 14]$ are tested. For each $x = [r; 14; s]$ record $\phi(x)$ and y the percentage of improvement achieved by optimizing this window ($h = 14$ comes from Bayesian optimization),

  Note: time to generate a single database $\approx$ 1-3 month.
- Sizes of the databases $\approx$ 3M schedules/windows,

Building the predictor: the databases

- To generate the *training*, *validation* and *test* databases, the same protocol has been used,
- For each database,
  1. Randomly generate 4000 instances of the scheduling problem for a given $n = 60$ and 2000 instances with $n = 100$,
  2. Run `MATH` in which all windows $[r; r + 14]$ are tested. For each $x = [r; 14; s]$ record $\phi(x)$ and y the percentage of improvement achieved by optimizing this window ($h = 14$ comes from Bayesian optimization),

  Note: time to generate a single database $\approx$ 1-3 month.
- Sizes of the databases $\approx$ 3M schedules/windows,

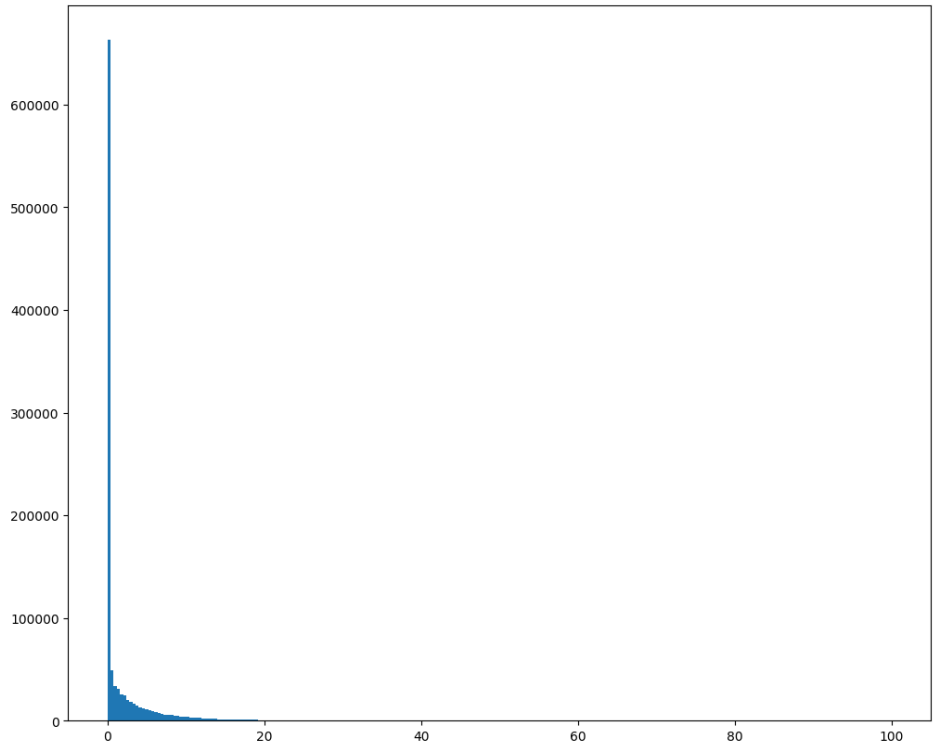
  Note: Databases must be filled with enough samples, representative of all cases and quite balanced.

Building the predictor: the databases

- To each row $x = [r; 14; s]$ in the database we associate $\phi(x)$ and y: $\phi(x)$ is normalized and the value y is normalized and multiplied by 100.

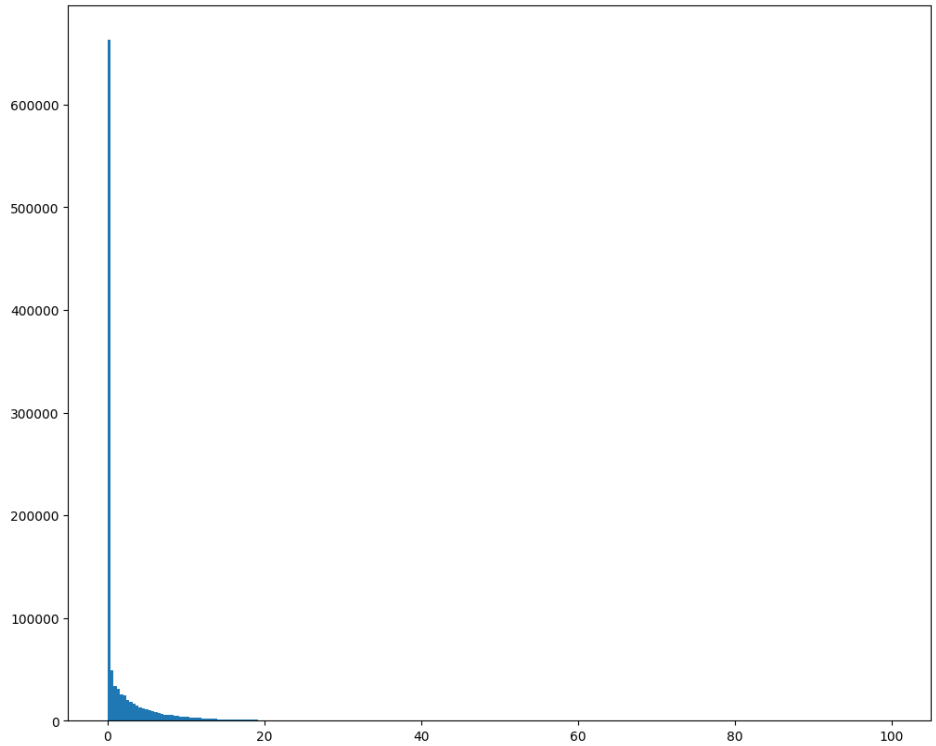
Building the predictor: the databases

- To each row $x = [r; 14; s]$ in the database we associate $\phi(x)$ and y: $\phi(x)$ is normalized and the value y is normalized and multiplied by 100.



Building the predictor: the databases

- To each row $x = [r; 14; s]$ in the database we associate $\phi(x)$ and y: $\phi(x)$ is normalized and the value y is normalized and multiplied by 100.



- A lot of "0%" improvements... but they are necessary to learn how to predict 0 (filtering of the databases doesn't help).

Building the predictor: tuning the learning process

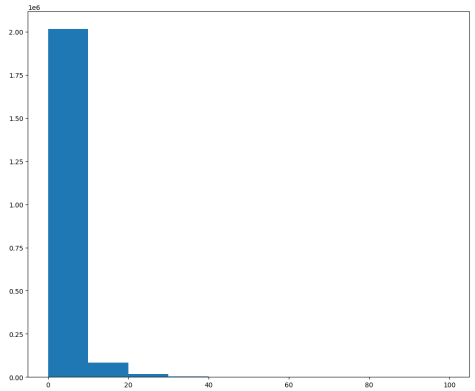
- Parameters of the learning process:

Building the predictor: tuning the learning process

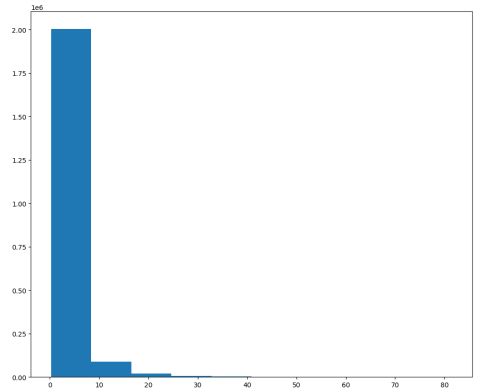
- Parameters of the learning process:
  1. Optimizer: ADAM,
  2. Batch size: 30000,
  3. Number of epochs: several runs of 8000, 4000, 4000 and 4000 epochs on the same model (restart),
  4. Learning rate λ: home-made procedure,
     1. $\lambda = 10^{-4}$ for the first 400 epochs,
     2. $\lambda = \lambda * 0.9995$ for the next 1100 epochs (final value $5.76.10^{-5}$),
     3. $\lambda = 2.10^{-5}$ for the next 1500 epochs,
     4. $\lambda = \lambda * 0.9995$ for the next epochs.
  5. Save the best model on the validation database (minimum loss),
  6. Shuffle the training database before each epoch,

Building the predictor: Results

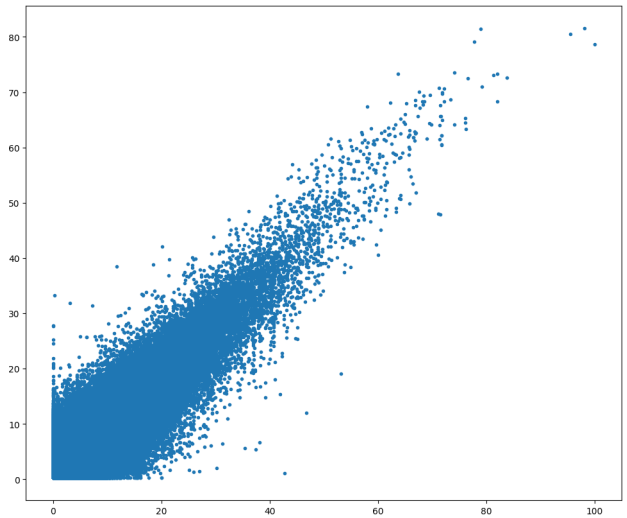
- On the training database: **mean error=-0.188 standard deviation=1.914**,



Labels to predict

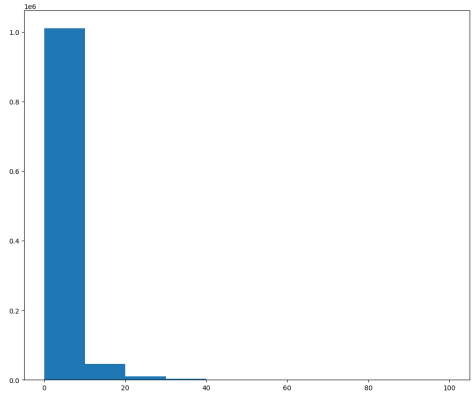


Predicted labels

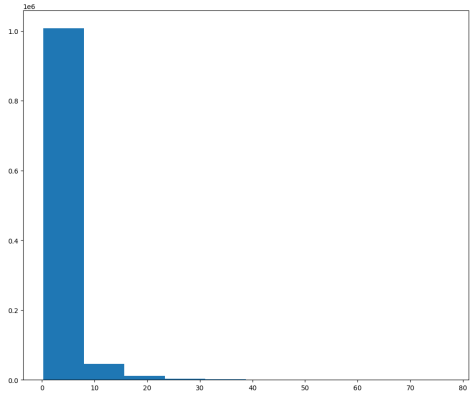


Building the predictor: Results

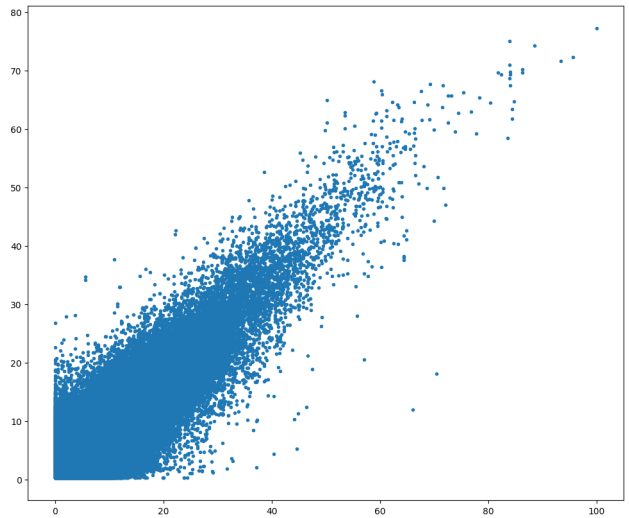
- On the validation database: **mean error=-0.005 standard deviation=2.362**,



Labels to predict

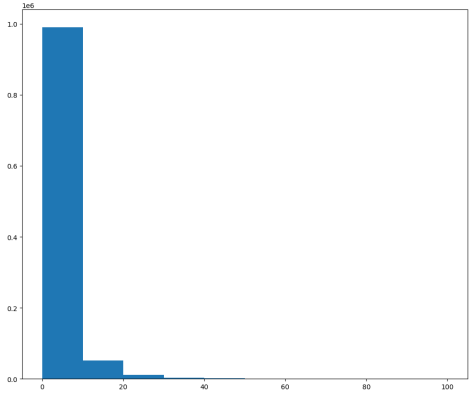


Predicted labels

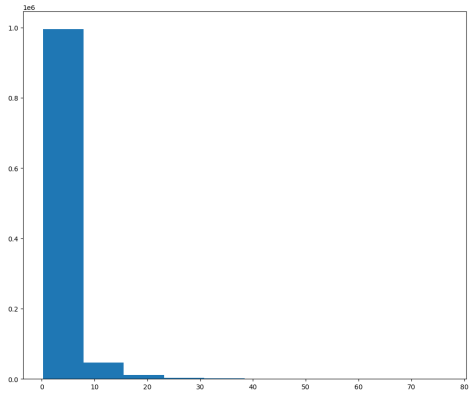


Building the predictor: Results

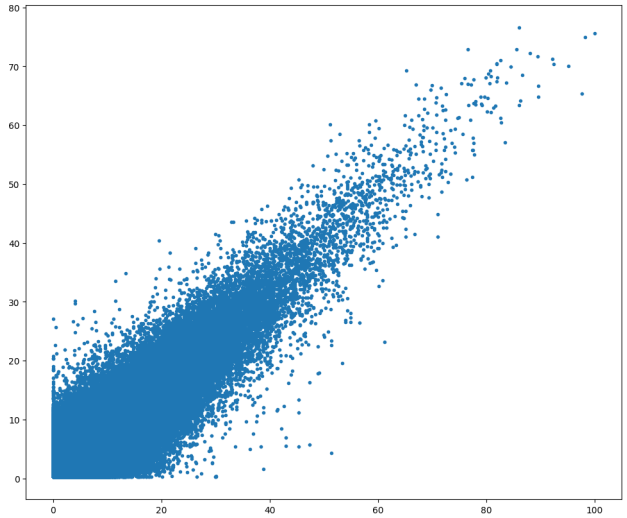
- On the test database: **mean error=0.175 standard deviation=2.629**,



Labels to predict



Predicted labels



Building the predictor: Results

- The predictor seems to be “good”...

Building the predictor: Results

- The predictor seems to be “good”...
- ... but the features do not capture well small improvements (around “0%”),

Building the predictor: Results

- The predictor seems to be “good”...
- ... but the features do not capture well small improvements (around “0%”),
- We tried other architecture sizes, restricted MLP (pyramidal shape)... not better,

Building the predictor: Results

- The predictor seems to be "good"...
- ... but the features do not capture well small improvements (around "0%"),
- We tried other architecture sizes, restricted MLP (pyramidal shape)... not better,
- So... let's test what happens when integrated to the matheuristic (ℓ-MATH)!

The ℓ-MATH heuristic

- How works ℓ-MATH?

The ℓ-MATH heuristic

- How works ℓ-MATH?
  1. At each iteration for a given s, generate all windows $[r; r + 14]$,

The ℓ-MATH heuristic

- How works ℓ-MATH?
  1. At each iteration for a given s, generate all windows $[r; r + 14]$,
  2. For each window $[r; r + 14]$, call the predictor and rank the windows by decreasing order of the predictions,

The ℓ-MATH heuristic

- How works ℓ-MATH?
  1. At each iteration for a given s, generate all windows $[r; r + 14]$,
  2. For each window $[r; r + 14]$, call the predictor and rank the windows by decreasing order of the predictions,
  3. Optimize the windows following this order until an improving one is found,

The ℓ-MATH heuristic

- How works ℓ-MATH?
  1. At each iteration for a given s, generate all windows $[r; r + 14]$,
  2. For each window $[r; r + 14]$, call the predictor and rank the windows by decreasing order of the predictions,
  3. Optimize the windows following this order until an improving one is found,
  4. If at a point the remaining windows all have a prediction < 4.5, then randomly select the next window within the remaining ones,

The ℓ-MATH heuristic

- How works ℓ-MATH?
  1. At each iteration for a given s, generate all windows $[r; r + 14]$,
  2. For each window $[r; r + 14]$, call the predictor and rank the windows by decreasing order of the predictions,
  3. Optimize the windows following this order until an improving one is found,
  4. If at a point the remaining windows all have a prediction < 4.5, then randomly select the next window within the remaining ones,
  5. Stops if a time limit is reached (same as for MATH) or no more improving windows.

Efficiency of ℓ-MATH

- We randomly generate instances and we compare four versions of the matheuristics:

Efficiency of ℓ-MATH

- We randomly generate instances and we compare four versions of the matheuristics:
  - MATH: the original matheuristic ($h = 12$),

Efficiency of ℓ-MATH

- We randomly generate instances and we compare four versions of the matheuristics:
  - MATH: the original matheuristic ($h = 12$),
  - MATH_{14}: a version of MATH with $h = 14$,

Efficiency of ℓ-MATH

- We randomly generate instances and we compare four versions of the matheuristics:
  - MATH: the original matheuristic ($h = 12$),
  - MATH_{14}: a version of MATH with $h = 14$,
  - b-MATH: the matheuristic with bayesian parameters,

Efficiency of ℓ-MATH

- We randomly generate instances and we compare four versions of the matheuristics:
  - MATH: the original matheuristic ($h = 12$),
  - MATH_{14}: a version of MATH with $h = 14$,
  - b-MATH: the matheuristic with bayesian parameters,
  - ℓ-MATH: decisions taken by the predictor and $h = 14$,

Efficiency of ℓ-MATH

- We randomly generate instances and we compare four versions of the matheuristics:
  - MATH: the original matheuristic ($h = 12$),
  - MATH_{14}: a version of MATH with $h = 14$,
  - b-MATH: the matheuristic with bayesian parameters,
  - ℓ-MATH: decisions taken by the predictor and $h = 14$,
- The tests have been done on a Intel Xeon Silver 4210R (8 cores at 2.40GHz) with 8Gb of RAM,

Efficiency of ℓ-MATH

- We randomly generate instances and we compare four versions of the matheuristics:
  - MATH: the original matheuristic ($h = 12$),
  - MATH_{14}: a version of MATH with $h = 14$,
  - b-MATH: the matheuristic with bayesian parameters,
  - ℓ-MATH: decisions taken by the predictor and $h = 14$,
- The tests have been done on a Intel Xeon Silver 4210R (8 cores at 2.40GHz) with 8Gb of RAM,
- Data generation protocol: $p_{j,i} \in U[10; 100], \forall j = 1..n, i = 1, 2,$

Efficiency of ℓ-MATH

- We randomly generate instances and we compare four versions of the matheuristics:
  - MATH: the original matheuristic ($h = 12$),
  - MATH_{14}: a version of MATH with $h = 14$,
  - b-MATH: the matheuristic with bayesian parameters,
  - ℓ-MATH: decisions taken by the predictor and $h = 14$,
- The tests have been done on a Intel Xeon Silver 4210R (8 cores at 2.40GHz) with 8Gb of RAM,
- Data generation protocol: $p_{j,i} \in U[10; 100], \forall j = 1..n, i = 1, 2$,
- For each problem size n, 50 instances are randomly generated,

Efficiency of ℓ-MATH

- We randomly generate instances and we compare four versions of the matheuristics:
  - MATH: the original matheuristic ($h = 12$),
  - MATH_{14}: a version of MATH with $h = 14$,
  - b-MATH: the matheuristic with bayesian parameters,
  - ℓ-MATH: decisions taken by the predictor and $h = 14$,
- The tests have been done on a Intel Xeon Silver 4210R (8 cores at 2.40GHz) with 8Gb of RAM,
- Data generation protocol: $p_{j,i} \in U[10; 100], \forall j = 1..n, i = 1, 2$,
- For each problem size n, 50 instances are randomly generated,
- On any given instance, each algorithm is ran 3 times and the results (time, $\sum_j C_j$ value) are averaged,

Efficiency of ℓ-MATH

n	MATH			MATH_{14}		
	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)
60	61.16	0.005	0.078	61.52	0.002	0.038
80	61.25	0.005	0.031	61.63	0.003	0.022
100	61.30	0.007	0.037	61.78	0.005	0.033
n	b-MATH			ℓ-MATH		
	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)
60	43.49	0.002	0.049	33.84	0.002	0.033
80	53.55	0.004	0.031	55.62	0.002	0.026
100	63.62	0.006	0.059	61.26	0.004	0.034

Efficiency of ℓ-MATH

n	MATH			MATH_{14}		
	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)
60	61.16	0.005	0.078	61.52	0.002	0.038
80	61.25	0.005	0.031	61.63	0.003	0.022
100	61.30	0.007	0.037	61.78	0.005	0.033
n	b-MATH			ℓ-MATH		
	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)
60	43.49	0.002	0.049	33.84	0.002	0.033
80	53.55	0.004	0.031	55.62	0.002	0.026
100	63.62	0.006	0.059	61.26	0.004	0.034

- b-MATH outperforms MATH (good!),

Efficiency of ℓ-MATH

n	MATH			MATH_{14}		
	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)
60	61.16	0.005	0.078	61.52	0.002	0.038
80	61.25	0.005	0.031	61.63	0.003	0.022
100	61.30	0.007	0.037	61.78	0.005	0.033
n	b-MATH			ℓ-MATH		
	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)
60	43.49	0.002	0.049	33.84	0.002	0.033
80	53.55	0.004	0.031	55.62	0.002	0.026
100	63.62	0.006	0.059	61.26	0.004	0.034

- b-MATH outperforms MATH (good!),
- MATH_{14} outperforms MATH and b-MATH,

Efficiency of ℓ-MATH

n	MATH			MATH_{14}		
	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)
60	61.16	0.005	0.078	61.52	0.002	0.038
80	61.25	0.005	0.031	61.63	0.003	0.022
100	61.30	0.007	0.037	61.78	0.005	0.033
n	b-MATH			ℓ-MATH		
	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)	t_{avg} (s)	δ_{avg} (%)	δ_{max} (%)
60	43.49	0.002	0.049	33.84	0.002	0.033
80	53.55	0.004	0.031	55.62	0.002	0.026
100	63.62	0.006	0.059	61.26	0.004	0.034

- b-MATH outperforms MATH (good!),
- MATH_{14} outperforms MATH and b-MATH,
- ℓ-MATH is the best heuristic on the average with a slightly reduced CPU time.

Outline

Conclusions

- Integrating ML into OR algorithms is a great research line!

Conclusions

- Integrating ML into OR algorithms is a great research line!
  - The two research works presented here show promising extensions,

Conclusions

- Integrating ML into OR algorithms is a great research line!
  - The two research works presented here show promising extensions,
  - The parameter tuning can be easily generalized,

Conclusions

- Integrating ML into OR algorithms is a great research line!
  - The two research works presented here show promising extensions,
  - The parameter tuning can be easily generalized,
  - There are perspectives to make the local search guiding oracle independent from features.

Conclusions

- Integrating ML into OR algorithms is a great research line!
  - The two research works presented here show promising extensions,
  - The parameter tuning can be easily generalized,
  - There are perspectives to make the local search guiding oracle independent from features.
- But keep your head on your shoulders!

Conclusions

- Integrating ML into OR algorithms is a great research line!
  - The two research works presented here show promising extensions,
  - The parameter tuning can be easily generalized,
  - There are perspectives to make the local search guiding oracle independent from features.
- But keep your head on your shoulders!
  - Generating "learning databases" is the key point: it is long and you may not generate the good ones at the first try,

Conclusions

- Integrating ML into OR algorithms is a great research line!
  - The two research works presented here show promising extensions,
  - The parameter tuning can be easily generalized,
  - There are perspectives to make the local search guiding oracle independent from features.
- But keep your head on your shoulders!
  - Generating "learning databases" is the key point: it is long and you may not generate the good ones at the first try,
  - Training times can also be long (though not the longest part),

Conclusions

- Integrating ML into OR algorithms is a great research line!
  - The two research works presented here show promising extensions,
  - The parameter tuning can be easily generalized,
  - There are perspectives to make the local search guiding oracle independent from features.
- But keep your head on your shoulders!
  - Generating "learning databases" is the key point: it is long and you may not generate the good ones at the first try,
  - Training times can also be long (though not the longest part),
  - Finally, you cannot do "everything with ML": beforehand, think about the time you need to train your predictors and see what is doable...