The marriage of Matheuristics and Scheduling

Vincent T'kindt

University of Tours,
LIFAT (EA 6300), France.

February 15, 2023

Outline

First contact

- MATHEuristics are not METAheuristics but are Metaheuristics,

[1a] Fischetti, M., Fischetti, M. (2018). Matheuristics. In: Martí R., Pardalos P., Resende M. (eds), Handbook of Heuristics. *Springer*.
[1b] Maniezzo, V., Stützle, T., Voß, S. (2009). Matheuristics: Hybridizing Metaheuristics and Mathematical Programming, 1st edn., *Springer*.
[1c] Ball, M.O. (2011). Heuristics based on mathematical programming, *Surveys in Operations Research and Management Science*, 16:21-38.
[2] Della Croce, F. (2016). MP or not MP: that is the question, *Journal of Scheduling*, 19:33-42.

First contact

- MATHEuristics are not METAheuristics but are Metaheuristics,
- General definition ([1a, 1b, 1c]):
  "Matheuristic is the hybridization of mathematical programming with metaheuristics. [...] Matheuristic is not a rigid paradigm but rather a concept framework for the design of mathematically sound heuristics."

[1a] Fischetti, M., Fischetti, M. (2018). Matheuristics. In: Martí R., Pardalos P., Resende M. (eds), Handbook of Heuristics. *Springer*.
[1b] Maniezzo, V., Stützle, T., Voß, S. (2009). Matheuristics: Hybridizing Metaheuristics and Mathematical Programming, 1st edn., *Springer*.
[1c] Ball, M.O. (2011). Heuristics based on mathematical programming, *Surveys in Operations Research and Management Science*, 16:21-38.
[2] Della Croce, F. (2016). MP or not MP: that is the question, *Journal of Scheduling*, 19:33-42.

First contact

- MATHEuristics are not METAheuristics but are Metaheuristics,
- General definition ([1a, 1b, 1c]):
  "Matheuristic is the hybridization of mathematical programming with metaheuristics. [...] Matheuristic is not a rigid paradigm but rather a concept framework for the design of mathematically sound heuristics."
- Take a scheduling problem and its MIP formulation, impose a time limit to the solver ⇒ matheuristic,

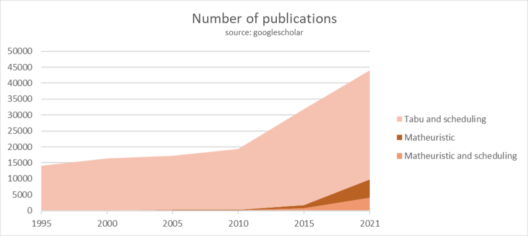
[1a] Fischetti, M., Fischetti, M. (2018). Matheuristics. In: Martí R., Pardalos P., Resende M. (eds), Handbook of Heuristics. *Springer*.
[1b] Maniezzo, V., Stützle, T., Voß, S. (2009). Matheuristics: Hybridizing Metaheuristics and Mathematical Programming, 1st edn., *Springer*.
[1c] Ball, M.O. (2011). Heuristics based on mathematical programming, *Surveys in Operations Research and Management Science*, 16:21-38.
[2] Della Croce, F. (2016). MP or not MP: that is the question, *Journal of Scheduling*, 19:33-42.

First contact

- MATHEuristics are not METAheuristics but are Metaheuristics,
- General definition ([1a, 1b, 1c]):
  "Matheuristic is the hybridization of mathematical programming with metaheuristics. [...] Matheuristic is not a rigid paradigm but rather a concept framework for the design of mathematically sound heuristics."
- Take a scheduling problem and its MIP formulation, impose a time limit to the solver $\Rightarrow$ matheuristic,
- Interest of Matheuristics: to rely on (more and more) efficient blackbox solvers ([2]),

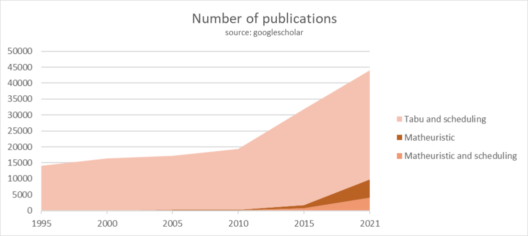
[1a] Fischetti, M., Fischetti, M. (2018). Matheuristics. In: Martí R., Pardalos P., Resende M. (eds), Handbook of Heuristics. *Springer*.
[1b] Maniezzo, V., Stützle, T., Voß, S. (2009). Matheuristics: Hybridizing Metaheuristics and Mathematical Programming, 1st edn., *Springer*.
[1c] Ball, M.O. (2011). Heuristics based on mathematical programming, *Surveys in Operations Research and Management Science*, 16:21-38.
[2] Della Croce, F. (2016). MP or not MP: that is the question, *Journal of Scheduling*, 19:33-42.

A quick look at the literature



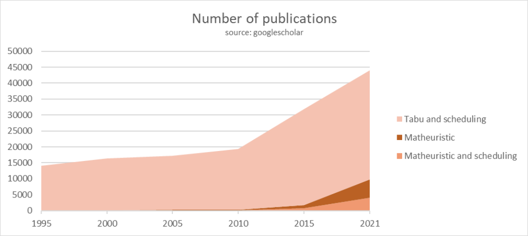
- Relatively recent,

A quick look at the literature



- Relatively recent,
- Mostly used for solving routing (and scheduling) problems,

A quick look at the literature



- Relatively recent,
- Mostly used for solving routing (and scheduling) problems,
- Hard to sketch a general scheme for matheuristics: RINS, Local Branching, VPLS, CMSA, Proximity Search, CRB, Relax-and-fix, POPMUSIC, ...

A quick look at the literature

- Constructive MH: use an IP to iteratively build a solution to the problem,

A quick look at the literature

- Constructive MH: use an IP to iteratively build a solution to the problem,
- Local Search MH: use an IP in the context of a local search (requires to know an initial solution),

A quick look at the literature

- Constructive MH: use an IP to iteratively build a solution to the problem,
- Local Search MH: use an IP in the context of a local search (requires to know an initial solution),
- Evolutionary MH: embed the solution of an IP into an evolutionary algorithm,

A quick look at the literature

- Constructive MH: use an IP to iteratively build a solution to the problem,
- Local Search MH: use an IP in the context of a local search (requires to know an initial solution),
- Evolutionary MH: embed the solution of an IP into an evolutionary algorithm,
- This talk: a personal view based on my own experience of Local Search MH.

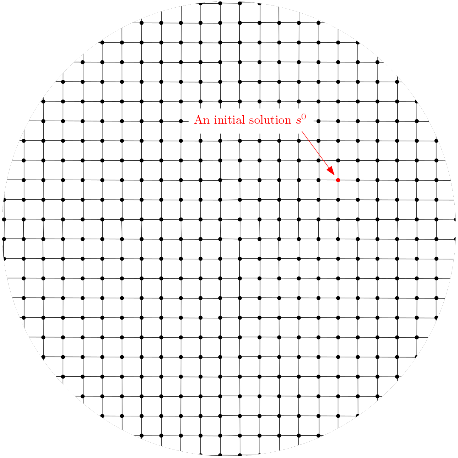
A general scheme (Local Search MH)

- 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.

A general scheme (Local Search MH)

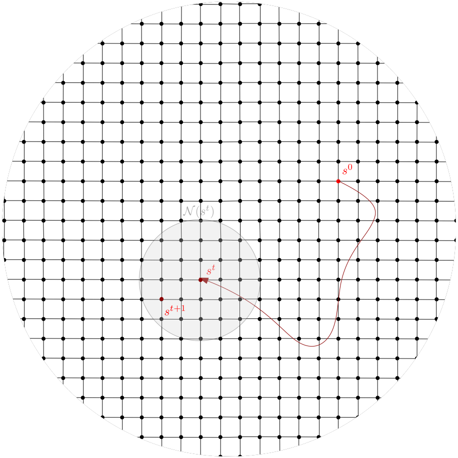
- Matheuristic as *LNS* heuristics ([1,3]),
- We start from a known initial solution s^0,



[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.

A general scheme (Local Search MH)

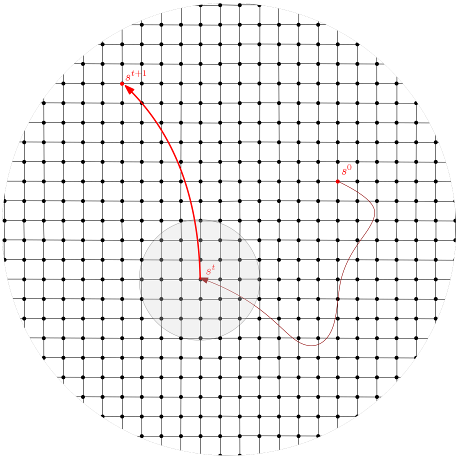
- Matheuristic as *LNS* heuristics ([1,3]),
- We start from a known initial solution s^0,
- Exploration of the neighbourhood $\mathcal{N}(s^t)$: by MIP (*intensification*),



[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.

A general scheme (Local Search MH)

- 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.

Outline

A general scheme (intensification)

- Consider a MIP formulation of your problem (**crucial choice**),

$$\min \sum_{j=1}^{n} C_{[j]} \tag{1}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1, \ldots, n \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1, \ldots, n \tag{3}$$

$$C_{[1]} = \sum_{i=1}^{n} (p_i + r_i) x_{i1} \tag{4}$$

$$C_{[j]} \geq C_{[j-1]} + \sum_{i=1}^{n} p_i x_{ij} \qquad \forall j = 2, \ldots, n \tag{5}$$

$$C_{[j]} \geq \sum_{i=1}^{n} (p_i + r_i) x_{ij} \qquad \forall j = 2, \ldots, n \tag{6}$$

$$x_{ij} \in \{0, 1\}, \quad C_{[j]} \geq 0 \tag{7}$$

$$\tag{8}$$

A general scheme (intensification)

- Consider a MIP formulation of your problem (**crucial choice**),
- Let be s^t the current solution and $x^t = [x_{ij}^t]_{ij}$ the associated values of variables,

$$\min \sum_{j=1}^{n} C_{[j]} \tag{1}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1, \ldots, n \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1, \ldots, n \tag{3}$$

$$C_{[1]} = \sum_{i=1}^{n} (p_i + r_i) x_{i1} \tag{4}$$

$$C_{[j]} \geq C_{[j-1]} + \sum_{i=1}^{n} p_i x_{ij} \quad \forall j = 2, \ldots, n \tag{5}$$

$$C_{[j]} \geq \sum_{i=1}^{n} (p_i + r_i) x_{ij} \quad \forall j = 2, \ldots, n \tag{6}$$

$$x_{ij} \in \{0, 1\}, \quad C_{[j]} \geq 0 \tag{7}$$

(8)

A general scheme (intensification)

- Consider a MIP formulation of your problem (**crucial choice**),
- Let be s^t the current solution and $x^t = [x^t_{ij}]_{ij}$ the associated values of variables,
- `Neighbourhood definition`: optimize *around* s^t allowing few variables x^t_{ij} to be changed,

$$\min \sum_{j=1}^{n} C_{[j]} \quad (1)$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1, \ldots, n \quad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1, \ldots, n \quad (3)$$

$$C_{[1]} = \sum_{i=1}^{n} (p_i + r_i) x_{i1} \quad (4)$$

$$C_{[j]} \geq C_{[j-1]} + \sum_{i=1}^{n} p_i x_{ij} \quad \forall j = 2, \ldots, n \quad (5)$$

$$C_{[j]} \geq \sum_{i=1}^{n} (p_i + r_i) x_{ij} \quad \forall j = 2, \ldots, n \quad (6)$$

$$x_{ij} \in \{0, 1\}, \quad C_{[j]} \geq 0 \quad (7)$$

$$(8)$$

A general scheme (intensification)

- Consider a MIP formulation of your problem (**crucial choice**),
- Let be s^t the current solution and $x^t = [x_{ij}^t]_{ij}$ the associated values of variables,
- `Neighbourhood definition`: optimize *around* s^t allowing few variables x_{ij}^t to be changed,
- Variable-fixing based intensification:

$$\min \sum_{j=1}^{n} C_{[j]} \quad (1)$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1, \ldots, n \quad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1, \ldots, n \quad (3)$$

$$C_{[1]} = \sum_{i=1}^{n} (p_i + r_i) x_{i1} \quad (4)$$

$$C_{[j]} \geq C_{[j-1]} + \sum_{i=1}^{n} p_i x_{ij} \quad \forall j = 2, \ldots, n \quad (5)$$

$$C_{[j]} \geq \sum_{i=1}^{n} (p_i + r_i) x_{ij} \quad \forall j = 2, \ldots, n \quad (6)$$

$$x_{ij} \in \{0, 1\}, \quad C_{[j]} \geq 0 \quad (7)$$

$$(8)$$

A general scheme (intensification)

- Consider a MIP formulation of your problem (**crucial choice**),
- Let be s^t the current solution and $x^t = [x^t_{ij}]_{ij}$ the associated values of variables,
- `Neighbourhood definition`: optimize *around* s^t allowing few variables x^t_{ij} to be changed,
- Variable-fixing based intensification:
  1. Determine a subset $\mathcal{S}^t$ of variables x_{ij},

$$\min \sum_{j=1}^{n} C_{[j]} \qquad (1)$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1, \ldots, n \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1, \ldots, n \qquad (3)$$

$$C_{[1]} = \sum_{i=1}^{n} (p_i + r_i) x_{i1} \qquad (4)$$

$$C_{[j]} \geq C_{[j-1]} + \sum_{i=1}^{n} p_i x_{ij} \qquad \forall j = 2, \ldots, n \qquad (5)$$

$$C_{[j]} \geq \sum_{i=1}^{n} (p_i + r_i) x_{ij} \qquad \forall j = 2, \ldots, n \qquad (6)$$

$$x_{ij} \in \{0, 1\}, \quad C_{[j]} \geq 0 \qquad (7)$$

$$(8)$$

A general scheme (intensification)

- Consider a MIP formulation of your problem (**crucial choice**),
- Let be s^t the current solution and $x^t = [x^t_{ij}]_{ij}$ the associated values of variables,
- `Neighbourhood definition`: optimize *around* s^t allowing few variables x^t_{ij} to be changed,
- Variable-fixing based intensification:
  1. Determine a subset $\mathcal{S}^t$ of variables x_{ij},
  2. Fix all variables in $\mathcal{S}^t$ to their value in x^t.

$$\min \sum_{j=1}^{n} C_{[j]} \quad (1)$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1, \ldots, n \quad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1, \ldots, n \quad (3)$$

$$C_{[1]} = \sum_{i=1}^{n} (p_i + r_i) x_{i1} \quad (4)$$

$$C_{[j]} \geq C_{[j-1]} + \sum_{i=1}^{n} p_i x_{ij} \qquad \forall j = 2, \ldots, n \quad (5)$$

$$C_{[j]} \geq \sum_{i=1}^{n} (p_i + r_i) x_{ij} \qquad \forall j = 2, \ldots, n \quad (6)$$

$$x_{ij} \in \{0, 1\}, \quad C_{[j]} \geq 0 \quad (7)$$

$$\mathbf{x_{ij} = x^t_{ij}} \qquad \forall \mathbf{x_{ij}} \in \mathcal{S}^\mathbf{t} \quad (8)$$

A general scheme (intensification)

- Variable-fixing based intensification: VPLS, Relaxation-Induced Neighbourhood Search (RINS), Fix-and-Optimize, ...

A general scheme (intensification)

- Variable-fixing based intensification: VPLS, Relaxation-Induced Neighbourhood Search (RINS), Fix-and-Optimize, ...
- Distance based intensification: local branching,

A general scheme (intensification)

- Variable-fixing based intensification: VPLS, Relaxation-Induced Neighbourhood Search (RINS), Fix-and-Optimize, ...
- Distance based intensification: local branching,
  1. Determine a subset $\mathcal{S}^t$ of variables x_{ij},
  2. Add a "distance measure" constraint, *e.g.* the Hamming distance:

$$\min \sum_{j=1}^{n} C_{[j]}$$

subject to

(1-7)

$$\Delta_{\mathcal{S}^t}(x, x^t) = \sum_{(ij)\in\mathcal{S}^t, x_{ij}^t=0} x_{ij} + \sum_{(ij)\in\mathcal{S}^t, x_{ij}^t=1} (1 - x_{ij}) \leq k$$

with k a given parameter.

VPLS: on which problem?

- We illustrate the Variable Partitioning Local Search (VPLS) on the $F2||\sum_j C_j$ problem ([4]),

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

VPLS: on which problem?

- We illustrate the Variable Partitioning Local Search (VPLS) on the $F2||\sum_j C_j$ problem ([4]),
  - n jobs have to be scheduled on two machines,

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

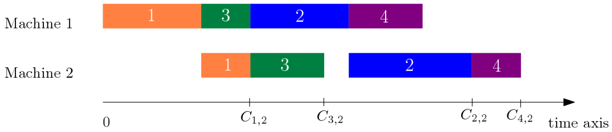
VPLS: on which problem?

- We illustrate the Variable Partitioning Local Search (VPLS) on the $F2||\sum_j C_j$ problem ([4]),
  - n jobs have to be scheduled on two machines,
  - Each job j is defined by a *processing time* $p_{j,i}$ on machine $i = 1, 2$,

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

VPLS: on which problem?

- We illustrate the Variable Partitioning Local Search (VPLS) on the $F2||\sum_j C_j$ problem ([4]),
  - 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,

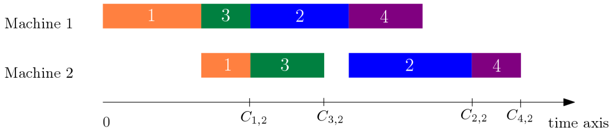


$C_{j,i}$: completion time of job j on machine i,

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

VPLS: on which problem?

- We illustrate the Variable Partitioning Local Search (VPLS) on the $F2||\sum_j C_j$ problem ([4]),
  - 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,



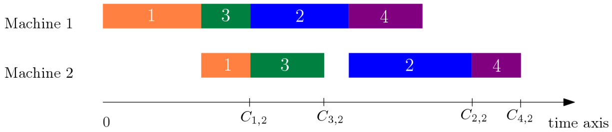
$C_{j,i}$: completion time of job j on machine i,

  - A schedule is a permutation σ of the jobs,

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

VPLS: on which problem?

- We illustrate the Variable Partitioning Local Search (VPLS) on the $F2||\sum_j C_j$ problem ([4]),
  - 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,
  - This problem is strongly $\mathcal{NP}$-hard.

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

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` $\mathcal{N}(s^t)$:

h=12

$$s^t = (1, 15, 3, 4, 2, 12, 8, 10, 6, 13, 9, 7, 11, 5, 14)$$

$$s^{t+1} = (1, 15, 3, 8, 4, 12, 2, 10, 6, 13, 9, 7, 11, 5, 14)$$

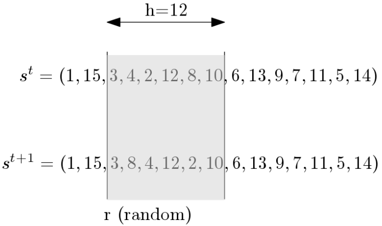
r (random)

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

[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` $\mathcal{N}(s^t)$:



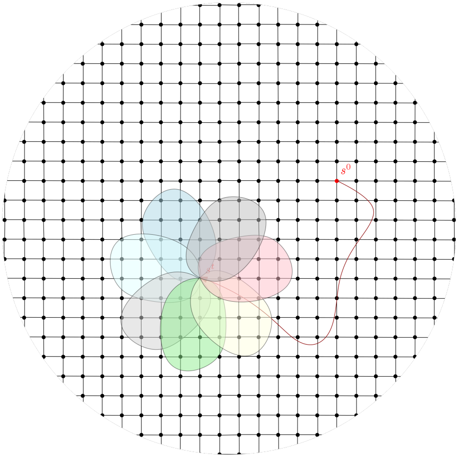
$$\mathcal{S}^t = \{x_{ij} | i = 1..r-1, r+h+1, ...n, j = 1..n\}$$

$\Rightarrow$ 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.

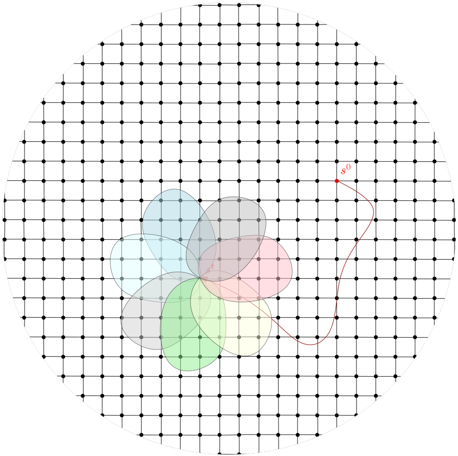
VPLS: the recipe

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



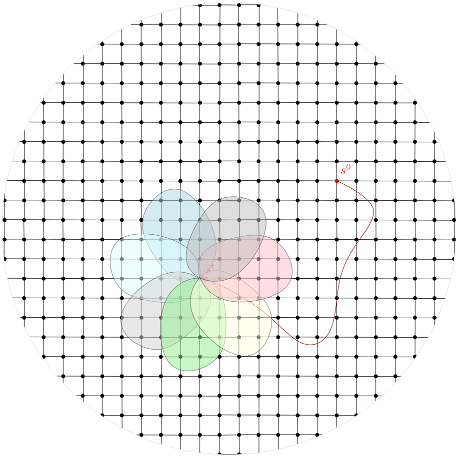
VPLS: the recipe

- 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.



VPLS: the cake

- Experimental results on randomly generated instances ([4]):

[6] Czapinski, M. (2010). Parallel simulated annealing with genetic enhancement for flowshop problem with Csum. *Computers & Industrial Engineering*, 59, 778–785.

VPLS: the cake

- Experimental results on randomly generated instances ([4]):
  - The choice of $h = 12$ is a good trade-off between time spent at each intensification phase and quality of the computed solution,

[6] Czapinski, M. (2010). Parallel simulated annealing with genetic enhancement for flowshop problem with Csum. *Computers & Industrial Engineering*, 59, 778–785.

VPLS: the cake

- Experimental results on randomly generated instances ([4]):
  - The choice of $h = 12$ is a good trade-off between time spent at each intensification phase and quality of the computed solution,
  - Results on instances with $n = 100$ ($T_{stop} = 60s$),

VPLS/LB (%)	CPLEX$_t$/LB (%)	VPLS/CPLEX$_t$ (%)	CPLEX$_t$-VPLS
0.26	0.46	0.20	370

[6] Czapinski, M. (2010). Parallel simulated annealing with genetic enhancement for flowshop problem with Csum. *Computers & Industrial Engineering*, 59, 778–785.

VPLS: the cake

- Experimental results on randomly generated instances ([4]):
  - The choice of $h = 12$ is a good trade-off between time spent at each intensification phase and quality of the computed solution,
  - Results on instances with $n = 100$ ($T_{stop} = 60s$),

VPLS/LB (%)	$\texttt{CPLEX}_t$/LB (%)	VPLS/$\texttt{CPLEX}_t$ (%)	$\texttt{CPLEX}_t$-VPLS
0.26	0.46	0.20	370

  - For $n = 300, 500$ similar results,

[6] Czapinski, M. (2010). Parallel simulated annealing with genetic enhancement for flowshop problem with Csum. *Computers & Industrial Engineering*, 59, 778–785.

VPLS: the cake

- Experimental results on randomly generated instances ([4]):
  - The choice of $h = 12$ is a good trade-off between time spent at each intensification phase and quality of the computed solution,
  - Results on instances with $n = 100$ ($T_{stop} = 60s$),

VPLS/LB (%)	CPLEX_t/LB (%)	VPLS/CPLEX_t (%)	CPLEX_t-VPLS
0.26	0.46	0.20	370

  - For $n = 300, 500$ similar results,
  - Best state-of-the-art heuristic for $n \leq 300$,

[6] Czapinski, M. (2010). Parallel simulated annealing with genetic enhancement for flowshop problem with Csum. *Computers & Industrial Engineering*, 59, 778–785.

VPLS: the cake

- Experimental results on randomly generated instances ([4]):
  - The choice of $h = 12$ is a good trade-off between time spent at each intensification phase and quality of the computed solution,
  - Results on instances with $n = 100$ ($T_{stop} = 60s$),

VPLS/LB (%)	CPLEX$_t$/LB (%)	VPLS/CPLEX$_t$ (%)	CPLEX$_t$-VPLS
0.26	0.46	0.20	370

  - For $n = 300, 500$ similar results,
  - Best state-of-the-art heuristic for $n \leq 300$,
  - Competitive with SAwGE ([6]) for $n = 500$ (due to computational requirement).

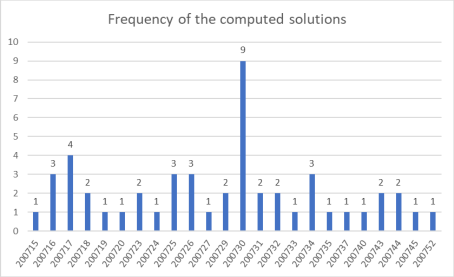
[6] Czapinski, M. (2010). Parallel simulated annealing with genetic enhancement for flowshop problem with Csum. *Computers & Industrial Engineering*, 59, 778–785.

VPLS: the cherry on the cake

- What about the impact of the random choice of r?

VPLS: the cherry on the cake

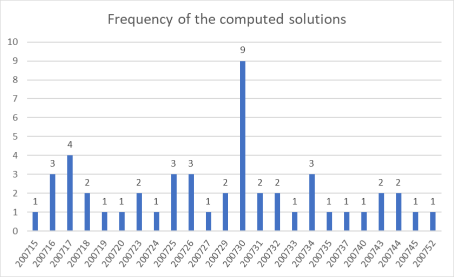
- What about the impact of the random choice of r?
- 50 executions of VPLS on an instance with $n = 100$ and $T_{stop} = 60s$,



Scale of values: [200715; 200752].

VPLS: the cherry on the cake

- What about the impact of the random choice of r?
- 50 executions of `VPLS` on an instance with $n = 100$ and $T_{stop} = 60s$,

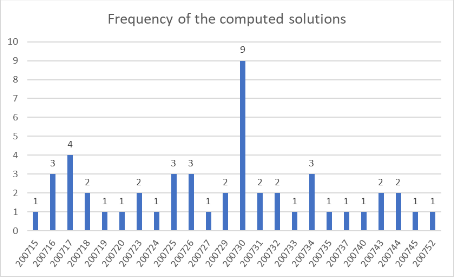


Scale of values: [200715; 200752].

- Time to best: $36s$ (avg) and $52s$ (max),

VPLS: the cherry on the cake

- What about the impact of the random choice of r?
- 50 executions of `VPLS` on an instance with $n = 100$ and $T_{stop} = 60s$,



Scale of values: [200715; 200752].

- Time to best: 36s (avg) and 52s (max),
- Improve the strategy for selecting a neighbourhood to explore or introduce *diversification*.

VPLS: Conclusions

- VPLS can be seen as a very efficient extension of standard local search heuristics,

VPLS: Conclusions

- VPLS can be seen as a very efficient extension of standard local search heuristics,
- The experiments on the $F2||\sum_j C_j$ problem highlight that:

VPLS: Conclusions

- VPLS can be seen as a very efficient extension of standard local search heuristics,
- The experiments on the $F2||\sum_j C_j$ problem highlight that:
    1. VPLS can be stuck in local optima,

VPLS: Conclusions

- VPLS can be seen as a very efficient extension of standard local search heuristics,
- The experiments on the $F2||\sum_j C_j$ problem highlight that:
  1. VPLS can be stuck in local optima,
  2. The random choice of r yields instability in terms of computed solution (alternative: test all r from 0 onwards),

VPLS: Conclusions

- VPLS can be seen as a very efficient extension of standard local search heuristics,
- The experiments on the $F2||\sum_j C_j$ problem highlight that:
  1. VPLS can be stuck in local optima,
  2. The random choice of r yields instability in terms of computed solution (alternative: test all r from 0 onwards),
  3. The experimental choice of h can be understood, but would not it be better to have a *dynamic* value of h?

VPLS: Conclusions

- VPLS can be seen as a very efficient extension of standard local search heuristics,
- The experiments on the $F2||\sum_j C_j$ problem highlight that:
  1. VPLS can be stuck in local optima,
  2. The random choice of r yields instability in terms of computed solution (alternative: test all r from 0 onwards),
  3. The experimental choice of h can be understood, but would not it be better to have a *dynamic* value of h?
  4. High CPU time: explore a lot of useless neighbourhoods.

VPLS: Conclusions

- VPLS can be seen as a very efficient extension of standard local search heuristics,
- The experiments on the $F2||\sum_j C_j$ problem highlight that:
  1. VPLS can be stuck in local optima,
  2. The random choice of r yields instability in terms of computed solution (alternative: test all r from 0 onwards),
  3. The experimental choice of h can be understood, but would not it be better to have a *dynamic* value of h?
  4. High CPU time: explore a lot of useless neighbourhoods.
- Considering windows of positions makes sense for permutation problems,

VPLS: Conclusions

- VPLS can be seen as a very efficient extension of standard local search heuristics,
- The experiments on the $F2||\sum_j C_j$ problem highlight that:
  1. VPLS can be stuck in local optima,
  2. The random choice of r yields instability in terms of computed solution (alternative: test all r from 0 onwards),
  3. The experimental choice of h can be understood, but would not it be better to have a *dynamic* value of h?
  4. High CPU time: explore a lot of useless neighbourhoods.
- Considering windows of positions makes sense for permutation problems,
- Can be extended to problems with assignment... but is it the best choice?

VPLS: Conclusions

- Use of distance based neighbourhood (case of the Hamming distance),

$$\Delta_{\mathcal{S}^t}(x, x^t) = \sum_{(ij)\in\mathcal{S}^t, x_{ij}^t=0} x_{ij} + \sum_{(ij)\in\mathcal{S}^t, x_{ij}^t=1} (1 - x_{ij})$$

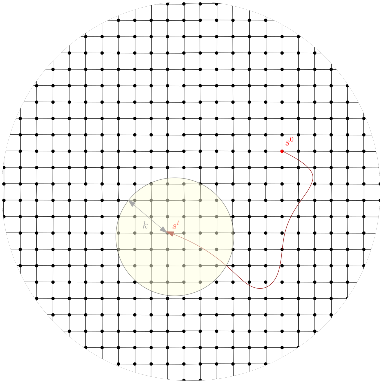
[7] Fischetti M., Monaci, M. (2014). Proximity search for 0–1 mixed-integer convex programming. *Journal of Heuristics*, 6(20):709–731.

VPLS: Conclusions

- Use of distance based neighbourhood (case of the Hamming distance),

$$\Delta_{\mathcal{S}^t}(x, x^t) = \sum_{(ij)\in\mathcal{S}^t, x^t_{ij}=0} x_{ij} + \sum_{(ij)\in\mathcal{S}^t, x^t_{ij}=1} (1 - x_{ij})$$

- `Neighbourhood definition`: $\mathcal{N}(s^t) = \{x | \Delta_{\mathcal{S}^t}(x, x^t) \leq k\}$, with k a given parameter,



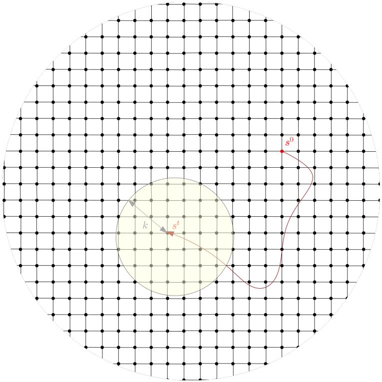
[7] Fischetti M., Monaci, M. (2014). Proximity search for 0–1 mixed-integer convex programming. *Journal of Heuristics*, 6(20):709–731.

VPLS: Conclusions

- Use of distance based neighbourhood (case of the Hamming distance),

$$\Delta_{\mathcal{S}^t}(x, x^t) = \sum_{(ij)\in\mathcal{S}^t, x_{ij}^t=0} x_{ij} + \sum_{(ij)\in\mathcal{S}^t, x_{ij}^t=1} (1 - x_{ij})$$

- `Neighbourhood definition`: $\mathcal{N}(s^t) = \{x | \Delta_{\mathcal{S}^t}(x, x^t) \leq k\}$, with k a given parameter,
- The neighbourhood is larger than with the windows of position approach $\Rightarrow$ intensification is expected to be more time consuming,



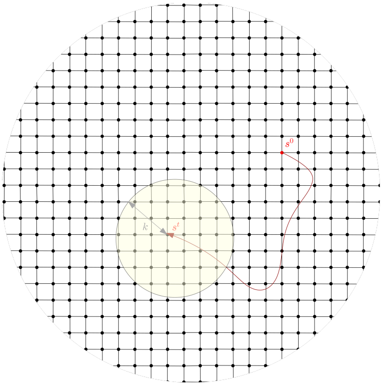
[7] Fischetti M., Monaci, M. (2014). Proximity search for 0–1 mixed-integer convex programming. *Journal of Heuristics*, 6(20):709–731.

VPLS: Conclusions

- Use of distance based neighbourhood (case of the Hamming distance),

$$\Delta_{\mathcal{S}^t}(x, x^t) = \sum_{(ij)\in\mathcal{S}^t, x_{ij}^t=0} x_{ij} + \sum_{(ij)\in\mathcal{S}^t, x_{ij}^t=1} (1 - x_{ij})$$

- `Neighbourhood definition`: $\mathcal{N}(s^t) = \{x | \Delta_{\mathcal{S}^t}(x, x^t) \leq k\}$, with k a given parameter,
- The neighbourhood is larger than with the windows of position approach $\Rightarrow$ intensification is expected to be more time consuming,
- VPLS with such a $\mathcal{N}(s^t)$ can be seen as a "dual" version of Proximity Search ([7]).



[7] Fischetti M., Monaci, M. (2014). Proximity search for 0–1 mixed-integer convex programming. *Journal of Heuristics*, 6(20):709–731.

Outline

A general scheme (diversification)

- *All you need is love... and diversification*,

A general scheme (diversification)

- *All you need is love... and diversification*,
- Diversification is just: *Get out of this neighbourhood!*
  $\Rightarrow$ Find a solution $x^{t+1} \notin \mathcal{N}(s^t)$.

A general scheme (diversification)

- *All you need is love... and diversification*,
- Diversification is just: *Get out of this neighbourhood!*
  $\Rightarrow$ Find a solution $x^{t+1} \notin \mathcal{N}(s^t)$.
- Easy to define on distance based neighbourhoods:

$$\text{Impose } \Delta_{\mathcal{S}^t}(x, x^t) > k.$$

A general scheme (diversification)

- *All you need is love... and diversification*,
- Diversification is just: *Get out of this neighbourhood!*
  $\Rightarrow$ Find a solution $x^{t+1} \notin \mathcal{N}(s^t)$.
- Easy to define on distance based neighbourhoods:

$$\text{Impose } \Delta_{\mathcal{S}^t}(x, x^t) > k.$$

- Local Branching is a perfect example of a matheuristic using both *intensification* and *diversification*.

Local Branching: the principles

- Consider the Hamming distance $\Delta_{\mathcal{S}^t}(x, x^t)$ on boolean variables x_{ij},

Local Branching: the principles

- Consider the Hamming distance $\Delta_{\mathcal{S}^t}(x, x^t)$ on boolean variables x_{ij},
- Introduced in [8] as an exact branching algorithm from which a matheuristic framework is derived,

Local Branching: the principles

- Consider the Hamming distance $\Delta_{\mathcal{S}^t}(x, x^t)$ on boolean variables x_{ij},
- Introduced in [8] as an exact branching algorithm from which a matheuristic framework is derived,
- We directly present Local Branching as a matheuristic assuming that:

Local Branching: the principles

- Consider the Hamming distance $\Delta_{\mathcal{S}^t}(x, x^t)$ on boolean variables x_{ij},
- Introduced in [8] as an exact branching algorithm from which a matheuristic framework is derived,
- We directly present Local Branching as a matheuristic assuming that:
  1. Each time a MIP has to be solved a time limit is imposed,

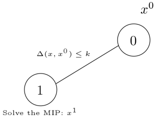
Local Branching: the principles

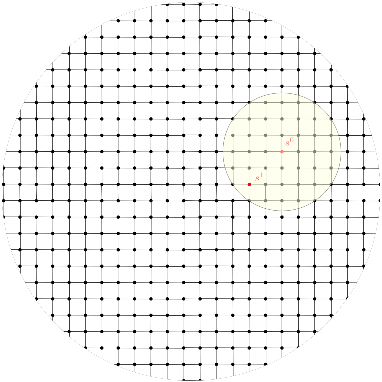
- Consider the Hamming distance $\Delta_{\mathcal{S}^t}(x, x^t)$ on boolean variables x_{ij},
- Introduced in [8] as an exact branching algorithm from which a matheuristic framework is derived,
- We directly present Local Branching as a matheuristic assuming that:
  1. Each time a MIP has to be solved a time limit is imposed,
  2. k is the given radius defining the size of the neighbourhood,

Local Branching: the principles

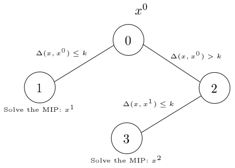
- Consider the Hamming distance $\Delta_{\mathcal{S}^t}(x, x^t)$ on boolean variables x_{ij},
- Introduced in [8] as an exact branching algorithm from which a matheuristic framework is derived,
- We directly present Local Branching as a matheuristic assuming that:
  1. Each time a MIP has to be solved a time limit is imposed,
  2. k is the given radius defining the size of the neighbourhood,
  3. ℓ is the given radius used for diversification.

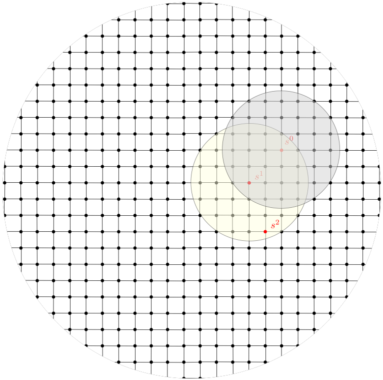
Local Branching: the principles



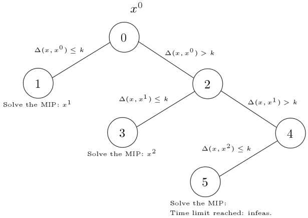


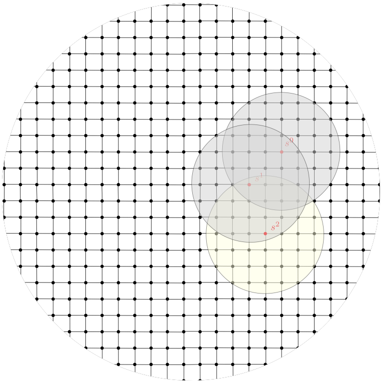
Local Branching: the principles



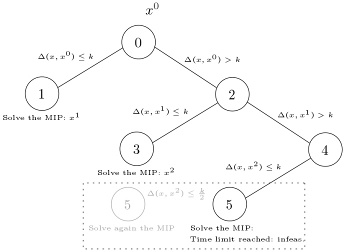


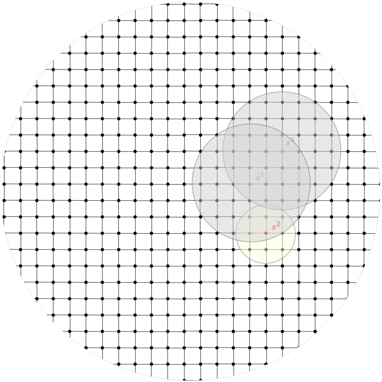
Local Branching: the principles



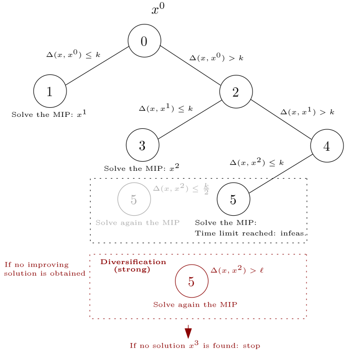


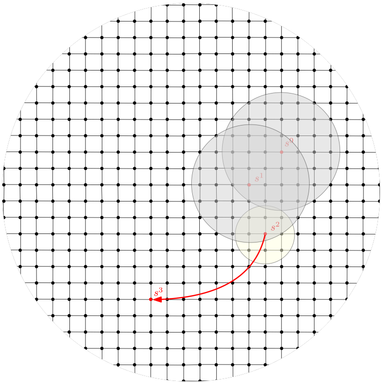
Local Branching: the principles



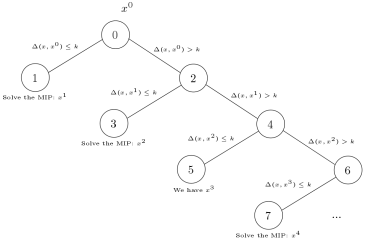


Local Branching: the principles

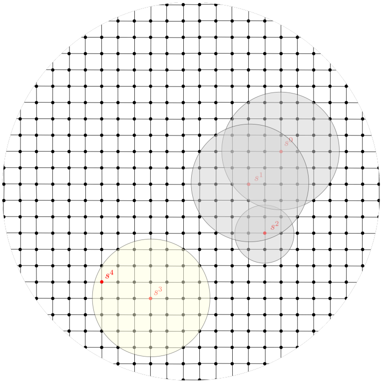




Local Branching: the principles



Stop after reaching a given time limit



Local Branching: Scheduling... you said Scheduling?

- Fischetti and Lodi ([8]) suggest:
  - To use a *soft* diversification before the strong one: try to find a feasible solution in a large neighbourhood, *e.g.* of size $\frac{3k}{2}$,

[9] Yang, F., Roel, L. (2021). Scheduling hybrid flow shops with time windows, *Journal of Heuristics*, 27:133–158.

Local Branching: Scheduling... you said Scheduling?

- Fischetti and Lodi ([8]) suggest:
  - To use a *soft* diversification before the strong one: try to find a feasible solution in a large neighbourhood, *e.g.* of size $\frac{3k}{2}$,
  - In the *strong* diversification to consider $\ell = 1$,

[9] Yang, F., Roel, L. (2021). Scheduling hybrid flow shops with time windows, *Journal of Heuristics*, 27:133–158.

Local Branching: Scheduling... you said Scheduling?

- Fischetti and Lodi ([8]) suggest:
  - To use a *soft* diversification before the strong one: try to find a feasible solution in a large neighbourhood, *e.g.* of size $\frac{3k}{2}$,
  - In the *strong* diversification to consider $\ell = 1$,
  - The matheuristic can be also stopped after a maximum number of diversifications is reached.

[9] Yang, F., Roel, L. (2021). Scheduling hybrid flow shops with time windows, *Journal of Heuristics*, 27:133–158.

Local Branching: Scheduling... you said Scheduling?

- Fischetti and Lodi ([8]) suggest:
  - To use a *soft* diversification before the strong one: try to find a feasible solution in a large neighbourhood, *e.g.* of size $\frac{3k}{2}$,
  - In the *strong* diversification to consider $\ell = 1$,
  - The matheuristic can be also stopped after a maximum number of diversifications is reached.
- Very (very) little applications on scheduling problems:

[9] Yang, F., Roel, L. (2021). Scheduling hybrid flow shops with time windows, *Journal of Heuristics*, 27:133–158.

Local Branching: Scheduling... you said Scheduling?

- Fischetti and Lodi ([8]) suggest:
  - To use a *soft* diversification before the strong one: try to find a feasible solution in a large neighbourhood, *e.g.* of size $\frac{3k}{2}$,
  - In the *strong* diversification to consider $\ell = 1$,
  - The matheuristic can be also stopped after a maximum number of diversifications is reached.
- Very (very) little applications on scheduling problems:
  1. In [9]: Applied to $FH2|r_j, \tilde{d}_j|\sum_j w_j C_j$. No diversification.

[9] Yang, F., Roel, L. (2021). Scheduling hybrid flow shops with time windows, *Journal of Heuristics*, 27:133–158.

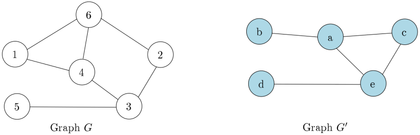
Local Branching: on which problem?

- Consider the following *Graph Edit Distance* (GED) problem,

[10] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). A local branching heuristic for solving a Graph Edit Distance problem, *Computers and Operations Research*, 106:225–235.

Local Branching: on which problem?

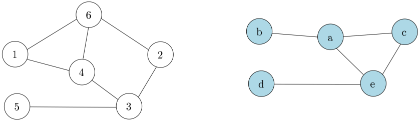
- Consider the following *Graph Edit Distance* (GED) problem,
  - We have two graphs $G = [V; E]$ and $G' = [V'; E']$...



[10] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). A local branching heuristic for solving a Graph Edit Distance problem, *Computers and Operations Research*, 106:225–235.

Local Branching: on which problem?

- Consider the following *Graph Edit Distance* (GED) problem,
  - We have two graphs $G = [V; E]$ and $G' = [V'; E']$...



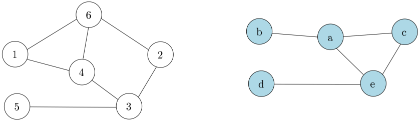
Graph G Graph G'

... and we want to know how much G resembles to G'.

[10] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). A local branching heuristic for solving a Graph Edit Distance problem, *Computers and Operations Research*, 106:225–235.

Local Branching: on which problem?

- Consider the following *Graph Edit Distance* (GED) problem,
  - We have two graphs $G = [V; E]$ and $G' = [V'; E']$...



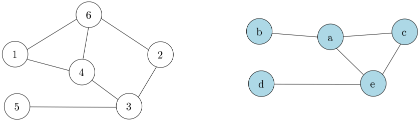
Graph G Graph G'

... and we want to know how much G resembles to G'.
  - In addition to the graphs, we know:

[10] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). A local branching heuristic for solving a Graph Edit Distance problem, *Computers and Operations Research*, 106:225–235.

Local Branching: on which problem?

- Consider the following *Graph Edit Distance* (GED) problem,
  - We have two graphs $G = [V; E]$ and $G' = [V'; E']$...



Graph G Graph G'

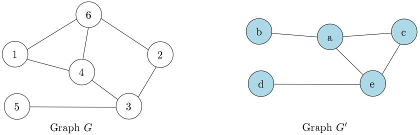
... and we want to know how much G resembles to G'.

  - In addition to the graphs, we know:
    1. The cost for deleting or creating a vertex from G: a constant τ,

[10] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). A local branching heuristic for solving a Graph Edit Distance problem, *Computers and Operations Research*, 106:225–235.

Local Branching: on which problem?

- Consider the following *Graph Edit Distance* (GED) problem,
  - We have two graphs $G = [V; E]$ and $G' = [V'; E']$...

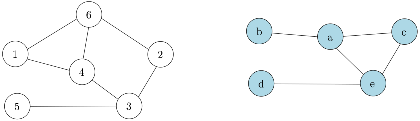


... and we want to know how much G resembles to G'.
  - In addition to the graphs, we know:
    1. The cost for deleting or creating a vertex from G: a constant τ,
    2. The cost for deleting or creating an edge in G: the same constant τ,

[10] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). A local branching heuristic for solving a Graph Edit Distance problem, *Computers and Operations Research*, 106:225–235.

Local Branching: on which problem?

- Consider the following *Graph Edit Distance* (GED) problem,
  - We have two graphs $G = [V; E]$ and $G' = [V'; E']$...



Graph G Graph G'

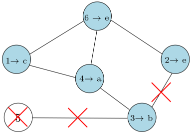
... and we want to know how much G resembles to G'.

  - In addition to the graphs, we know:
    1. The cost for deleting or creating a vertex from G: a constant τ,
    2. The cost for deleting or creating an edge in G: the same constant τ,
    3. The cost c_{jk} for matching $j \in V$ with $k \in V'$.

[10] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). A local branching heuristic for solving a Graph Edit Distance problem, *Computers and Operations Research*, 106:225–235.

Local Branching: on which problem?

- Consider the following *Graph Edit Distance* (GED) problem,
  - We have two graphs $G = [V; E]$ and $G' = [V'; E']$...



A matching

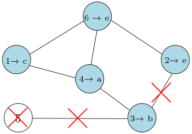
... and we want to know how much G resembles to G'.

  - In addition to the graphs, we know:
    1. The cost for deleting or creating a vertex from G: a constant τ,
    2. The cost for deleting or creating an edge in G: the same constant τ,
    3. The cost c_{jk} for matching $j \in V$ with $k \in V'$.
  - Find the matching of G and G' which minimizes the total matching cost,

[10] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). A local branching heuristic for solving a Graph Edit Distance problem, *Computers and Operations Research*, 106:225–235.

Local Branching: on which problem?

- Consider the following *Graph Edit Distance* (GED) problem,
  - We have two graphs $G = [V; E]$ and $G' = [V'; E']$...



A matching

... and we want to know how much G resembles to G'.
  - In addition to the graphs, we know:
    1. The cost for deleting or creating a vertex from G: a constant τ,
    2. The cost for deleting or creating an edge in G: the same constant τ,
    3. The cost c_{jk} for matching $j \in V$ with $k \in V'$.
  - Find the matching of G and G' which minimizes the total matching cost,
  - This problem is strongly $\mathcal{NP}$-hard.

[10] Darwiche, M., Conte, D., Raveaux, R., T'kindt, V. (2019). A local branching heuristic for solving a Graph Edit Distance problem, *Computers and Operations Research*, 106:225–235.

Local Branching on the GED: the ingredients

- This is an assignment problem for which we use the following IP formulation ([11]),

$$\text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{N}\left(c_{ij}x_{ij} + \frac{\tau}{2}(s_{ij} + t_{ij})\right)$$

st

$$\sum_{k=1}^{N} A_{ik}x_{kj} - \sum_{c=1}^{N} x_{ic}A'_{cj} + s_{ij} - t_{ij} = 0 \qquad \forall i, j = 1..N$$

$$\sum_{i=1}^{N} x_{ik} = \sum_{j=1}^{N} x_{kj} = 1 \qquad \forall k = 1..N$$

[11] Justice, D. , Hero, A. (2006). A binary linear programming formulation of the graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1200–1214.

Local Branching on the GED: the ingredients

- This is an assignment problem for which we use the following IP formulation ([11]),

- Boolean variables $x_{ij} = 1$ if vertex $i \in V$ is matched with vertex $j \in V'$,

$$\text{Minimize} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(c_{ij} x_{ij} + \frac{\tau}{2}(s_{ij} + t_{ij}) \right)$$

st

$$\sum_{k=1}^{N} A_{ik} x_{kj} - \sum_{c=1}^{N} x_{ic} A'_{cj} + s_{ij} - t_{ij} = 0 \qquad \forall i, j = 1..N$$

$$\sum_{i=1}^{N} x_{ik} = \sum_{j=1}^{N} x_{kj} = 1 \qquad \forall k = 1..N$$

[11] Justice, D. , Hero, A. (2006). A binary linear programming formulation of the graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1200–1214.

Local Branching on the GED: the ingredients

- This is an assignment problem for which we use the following IP formulation ([11]),

- Boolean variables $x_{ij} = 1$ if vertex $i \in V$ is matched with vertex $j \in V'$,
- Boolean variables s_{ij} and t_{ij} represent edge deletion/adding,

$$\text{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{N}\left(c_{ij}x_{ij} + \frac{\tau}{2}(s_{ij} + t_{ij})\right)$$

st

$$\sum_{k=1}^{N} A_{ik}x_{kj} - \sum_{c=1}^{N} x_{ic}A'_{cj} + s_{ij} - t_{ij} = 0 \qquad \forall i, j = 1..N$$

$$\sum_{i=1}^{N} x_{ik} = \sum_{j=1}^{N} x_{kj} = 1 \qquad \forall k = 1..N$$

[11] Justice, D. , Hero, A. (2006). A binary linear programming formulation of the graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1200–1214.

Local Branching on the GED: the ingredients

- This is an assignment problem for which we use the following IP formulation ([11]),

- Boolean variables $x_{ij} = 1$ if vertex $i \in V$ is matched with vertex $j \in V'$,
- Boolean variables s_{ij} and t_{ij} represent edge deletion/adding,
- The important variables are the x_{ij}'s $\Rightarrow$ $\mathcal{S}^t = \{x_{ij} | i, j = 1..N\}$.

$$\text{Minimize} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(c_{ij} x_{ij} + \frac{\tau}{2} (s_{ij} + t_{ij}) \right)$$

st

$$\sum_{k=1}^{N} A_{ik} x_{kj} - \sum_{c=1}^{N} x_{ic} A'_{cj} + s_{ij} - t_{ij} = 0 \qquad \forall i, j = 1..N$$

$$\sum_{i=1}^{N} x_{ik} = \sum_{j=1}^{N} x_{kj} = 1 \qquad \forall k = 1..N$$

[11] Justice, D. , Hero, A. (2006). A binary linear programming formulation of the graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1200–1214.

Local Branching on the GED: the ingredients

- `Initialization:` x^0 is computed by solving the IP with a (short) time limit,

Local Branching on the GED: the ingredients

- `Initialization:` x^0 is computed by solving the IP with a (short) time limit,
- `Neighbourhood definition:` $\mathcal{N}(s^t) = \{x|\Delta_{\mathcal{S}^t}(x, x^t) \leq k\}$,
- `Intensification:`
  1. Explore $\mathcal{N}(s^t)$ by solving the IP within a time limit T_{node},

Local Branching on the GED: the ingredients

- `Initialization:` x^0 is computed by solving the IP with a (short) time limit,
- `Neighbourhood definition:` $\mathcal{N}(s^t) = \{x | \Delta_{\mathcal{S}^t}(x, x^t) \leq k\}$,
- `Intensification:`
  1. Explore $\mathcal{N}(s^t)$ by solving the IP within a time limit T_{node},
  2. If no improving solutions are found due to the time limit $\Rightarrow$ change the radius to $\frac{k}{2}$ and solve again.

Local Branching on the GED: the ingredients

- `Initialization:` x^0 is computed by solving the IP with a (short) time limit,
- `Neighbourhood definition:` $\mathcal{N}(s^t) = \{x | \Delta_{\mathcal{S}^t}(x, x^t) \leq k\}$,
- `Intensification:`
  1. Explore $\mathcal{N}(s^t)$ by solving the IP within a time limit T_{node},
  2. If no improving solutions are found due to the time limit $\Rightarrow$ change the radius to $\frac{k}{2}$ and solve again.
- `Diversification:`

Local Branching on the GED: the ingredients

- `Initialization:` x^0 is computed by solving the IP with a (short) time limit,
- `Neighbourhood definition:` $\mathcal{N}(s^t) = \{x | \Delta_{\mathcal{S}^t}(x, x^t) \leq k\}$,
- `Intensification:`
    1. Explore $\mathcal{N}(s^t)$ by solving the IP within a time limit T_{node},
    2. If no improving solutions are found due to the time limit $\Rightarrow$ change the radius to $\frac{k}{2}$ and solve again.
- `Diversification:`
    1. *Soft diversification* doesn't help,

Local Branching on the GED: the ingredients

- `Initialization:` x^0 is computed by solving the IP with a (short) time limit,
- `Neighbourhood definition:` $\mathcal{N}(s^t) = \{x | \Delta_{\mathcal{S}^t}(x, x^t) \leq k\}$,
- `Intensification:`
  1. Explore $\mathcal{N}(s^t)$ by solving the IP within a time limit T_{node},
  2. If no improving solutions are found due to the time limit $\Rightarrow$ change the radius to $\frac{k}{2}$ and solve again.
- `Diversification:`
  1. *Soft diversification* doesn't help,
  2. *Strong diversification* on a subset $\mathcal{S}_I^t \subseteq \mathcal{S}^t$ of "important" variables,

Local Branching on the GED: the ingredients

- `Diversification:`
  1. How to determine $\mathcal{S}_l^t$?

Local Branching on the GED: the ingredients

- Diversification:
  1. How to determine $\mathcal{S}_I^t$?
  2. Independently of x^t, compute matrix $M = [m_{ij}]_{ij}$ with m_{ij} the cost of matching vertex $i \in V$ with $j \in V'$,

$$m_{ij} = c_{ij} + \theta_{ij},$$

  with θ_{ij} a lower bound on the costs induced by matching the edges of i with those of j (assignment problem).

Local Branching on the GED: the ingredients

- Diversification:
  1. How to determine $\mathcal{S}_I^t$?
  2. Independently of x^t, compute matrix $M = [m_{ij}]_{ij}$ with m_{ij} the cost of matching vertex $i \in V$ with $j \in V'$,
  $$m_{ij} = c_{ij} + \theta_{ij},$$
  with θ_{ij} a lower bound on the costs induced by matching the edges of i with those of j (assignment problem).
  3. We compute standard deviations σ_i over all values $m_{ij}, \forall j = 1..|V'| + 1$,

Local Branching on the GED: the ingredients

- `Diversification:`
  1. How to determine $\mathcal{S}_I^t$?
  2. Independently of x^t, compute matrix $M = [m_{ij}]_{ij}$ with m_{ij} the cost of matching vertex $i \in V$ with $j \in V'$,
  $$m_{ij} = c_{ij} + \theta_{ij},$$
  with θ_{ij} a lower bound on the costs induced by matching the edges of i with those of j (assignment problem).
  3. We compute standard deviations σ_i over all values $m_{ij}, \forall j = 1..|V'| + 1$,
  4. Binary classification of the variables (Nearest Neighbour) to separate the small from the high standard deviation vertices,

Local Branching on the GED: the ingredients

- `Diversification:`
  1. How to determine $\mathcal{S}_I^t$?
  2. Independently of x^t, compute matrix $M = [m_{ij}]_{ij}$ with m_{ij} the cost of matching vertex $i \in V$ with $j \in V'$,
  $$m_{ij} = c_{ij} + \theta_{ij},$$
  with θ_{ij} a lower bound on the costs induced by matching the edges of i with those of j (assignment problem).
  3. We compute standard deviations σ_i over all values $m_{ij}, \forall j = 1..|V'| + 1$,
  4. Binary classification of the variables (Nearest Neighbour) to separate the small from the high standard deviation vertices,
  5. $\mathcal{S}_I^t$ contains the variables x_{ij} associated to the the high standard deviation vertices $i \in V$,

Local Branching on the GED: the ingredients

- `Diversification:`
  1. How to determine $\mathcal{S}_I^t$?
  2. Independently of x^t, compute matrix $M = [m_{ij}]_{ij}$ with m_{ij} the cost of matching vertex $i \in V$ with $j \in V'$,
  $$m_{ij} = c_{ij} + \theta_{ij},$$
  with θ_{ij} a lower bound on the costs induced by matching the edges of i with those of j (assignment problem).
  3. We compute standard deviations σ_i over all values $m_{ij}, \forall j = 1..|V'| + 1$,
  4. Binary classification of the variables (Nearest Neighbour) to separate the small from the high standard deviation vertices,
  5. $\mathcal{S}_I^t$ contains the variables x_{ij} associated to the the high standard deviation vertices $i \in V$,
  6. To diversify with solve the IP with the constraint:
  $$\Delta_{\mathcal{S}_I^t}(x, x^t) \geq \ell.$$

Local Branching on the GED: the cake

- Stopping condition: (max time limit T_{solve} is reached) or (max number of diversifications Div_{solve} is reached),

Local Branching on the GED: the cake

- Stopping condition: (max time limit T_{solve} is reached) or (max number of diversifications Div_{solve} is reached),
- Experimental results on two databases of graphs representing chemical molecules,

Local Branching on the GED: the cake

- Stopping condition: (max time limit T_{solve} is reached) or (max number of diversifications Div_{solve} is reached),
- Experimental results on two databases of graphs representing chemical molecules,
  - PAH: 94 graphs with up to 28 vertices (8836 instances).
    - $k = 20$, $\ell = 30$,
    - $T_{node} = 1.75s$, $T_{solve} = 12.25s$, $Div_{solve} = 3$.

Local Branching on the GED: the cake

- Stopping condition: (max time limit T_{solve} is reached) or (max number of diversifications Div_{solve} is reached),
- Experimental results on two databases of graphs representing chemical molecules,
  - PAH: 94 graphs with up to 28 vertices (8836 instances).
    - $k = 20$, $\ell = 30$,
    - $T_{node} = 1.75s$, $T_{solve} = 12.25s$, $Div_{solve} = 3$.
  - MUTA: 80 graphs from 10 to 70 vertices (6400 instances).
    - $k = 20$, $\ell = 30$,
    - $T_{node} = 180s$, $T_{solve} = 900s$, $Div_{solve} = 3$.

Local Branching on the GED: the cake

- On PAH instances:

[1] Computed by solving the IP formulation with a time limit of 10h per instance

Local Branching on the GED: the cake

- On PAH instances:
  - Average CPU time: 3s,

[1] Computed by solving the IP formulation with a time limit of 10h per instance

Local Branching on the GED: the cake

- On PAH instances:
  - Average CPU time: 3s,
  - Gap to optimality: $< 0.35\%$,

[1] Computed by solving the IP formulation with a time limit of 10h per instance

Local Branching on the GED: the cake

- On PAH instances:
  - Average CPU time: 3s,
  - Gap to optimality: $< 0.35\%$,
  - 76% of the instances were solved to optimality by local branching.

[1] Computed by solving the IP formulation with a time limit of 10h per instance

Local Branching on the GED: the cake

- On PAH instances:
  - Average CPU time: 3s,
  - Gap to optimality: $< 0.35\%$,
  - 76% of the instances were solved to optimality by local branching.
- On MUTA instances:

[1] Computed by solving the IP formulation with a time limit of 10h per instance

Local Branching on the GED: the cake

- On PAH instances:
  - Average CPU time: 3s,
  - Gap to optimality: $< 0.35\%$,
  - 76% of the instances were solved to optimality by local branching.
- On MUTA instances:
  - Average CPU time: 750s on the largest instances,

[1] Computed by solving the IP formulation with a time limit of 10h per instance

Local Branching on the GED: the cake

- On PAH instances:
  - Average CPU time: 3s,
  - Gap to optimality: $< 0.35\%$,
  - 76% of the instances were solved to optimality by local branching.
- On MUTA instances:
  - Average CPU time: 750s on the largest instances,
  - Gap to the best known solution[1]: $< 0.78\%$.

$\Rightarrow$ Outperforms all the known heuristics (in 2021) on the GED problem.

[1] Computed by solving the IP formulation with a time limit of 10h per instance

Matheuristics: a first pit stop

- Very efficient heuristics...

Matheuristics: a first pit stop

- Very efficient heuristics...
- Some points of attention when designing such a heuristic:

Matheuristics: a first pit stop

- Very efficient heuristics...
- Some points of attention when designing such a heuristic:
  1. The choice of the IP formulation is crucial: fast convergence towards an optimal solution,

Matheuristics: a first pit stop

- Very efficient heuristics...
- Some points of attention when designing such a heuristic:
  1. The choice of the IP formulation is crucial: fast convergence towards an optimal solution,
  2. Not necessary to let the IP solver running until it proves optimality: experimental tuning,

Matheuristics: a first pit stop

- Very efficient heuristics...
- Some points of attention when designing such a heuristic:
  1. The choice of the IP formulation is crucial: fast convergence towards an optimal solution,
  2. Not necessary to let the IP solver running until it proves optimality: experimental tuning,
  3. The choice of variables $\mathcal{S}^t$ is important: put the variables generating the combinatorics (and maybe not all of them),

Matheuristics: a first pit stop

- Very efficient heuristics...
- Some points of attention when designing such a heuristic:
  1. The choice of the IP formulation is crucial: fast convergence towards an optimal solution,
  2. Not necessary to let the IP solver running until it proves optimality: experimental tuning,
  3. The choice of variables $\mathcal{S}^t$ is important: put the variables generating the combinatorics (and maybe not all of them),
  4. Neighbourhood size (r, h, k...): must be fixed to find a good tradeoff between minimizing the number of iterations and total CPU time,

Matheuristics: a first pit stop

- Very efficient heuristics...
- Some points of attention when designing such a heuristic:
  1. The choice of the IP formulation is crucial: fast convergence towards an optimal solution,
  2. Not necessary to let the IP solver running until it proves optimality: experimental tuning,
  3. The choice of variables $\mathcal{S}^t$ is important: put the variables generating the combinatorics (and maybe not all of them),
  4. Neighbourhood size (r, h, k...): must be fixed to find a good tradeoff between minimizing the number of iterations and total CPU time,
  5. Diversification seems to be really useful.

Matheuristics: a first pit stop

- ... but time consuming!

Matheuristics: a first pit stop

- ... but time consuming!
- Ways to improve the situation:

Matheuristics: a first pit stop

- ... but time consuming!
- Ways to improve the situation:
  1. Limit the number of “useless” neighbourhoods (VPLS),

Matheuristics: a first pit stop

- ... but time consuming!
- Ways to improve the situation:
  1. Limit the number of "useless" neighbourhoods (VPLS),
  2. Limit the size of the IPs to solve: time and ability to solve large-size instances (VPLS more suitable than Local Branching),

Matheuristics: a first pit stop

- ... but time consuming!
- Ways to improve the situation:
  1. Limit the number of "useless" neighbourhoods (VPLS),
  2. Limit the size of the IPs to solve: time and ability to solve large-size instances (VPLS more suitable than Local Branching),
  3. Adjust the neighbourhood size dynamically.

Outline

Matheuristics and Machine Learning

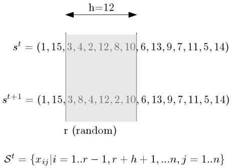
- Goal: find the most suitable neighbourhoods to explore,

Matheuristics and Machine Learning

- Goal: find the most suitable neighbourhoods to explore,
- We go back to the $F2||\sum_j C_j$ problem and VPLS,

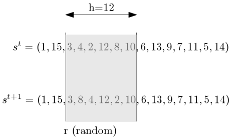
Matheuristics and Machine Learning

- Goal: find the most suitable neighbourhoods to explore,
- We go back to the $F2||\sum_j C_j$ problem and VPLS,
- `Neighbourhood definition` $\mathcal{N}(s^t)$:



Matheuristics and Machine Learning

- Goal: find the most suitable neighbourhoods to explore,
- We go back to the $F2||\sum_j C_j$ problem and VPLS,
- `Neighbourhood definition` $\mathcal{N}(s^t)$:

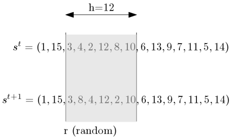


$$\mathcal{S}^t = \{x_{ij} | i = 1..r-1, r+h+1, ...n, j = 1..n\}$$

- The neighbourhoods to explore are defined by r and h,

Matheuristics and Machine Learning

- Goal: find the most suitable neighbourhoods to explore,
- We go back to the $F2||\sum_j C_j$ problem and VPLS,
- `Neighbourhood definition` $\mathcal{N}(s^t)$:



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

- The neighbourhoods to explore are defined by r and h,
- Can we use Machine Learning to predict the best r and h for a given s^t?

The ml-VPLS heuristic

- Ideal goal: to have an oracle (predictor) capable of predicting the values of r and h for a given s^t,

[40] T'kindt, V., Raveaux, R. (2022). A learning based matheuristic to solve the two machine flowshop scheduling problem with sum of completion times. *23rd French Conference on Operations Research and Decision Aid (ROADEF)*, Lyon, France.

The ml-VPLS heuristic

- Ideal goal: to have an oracle (predictor) capable of predicting the values of r and h for a given s^t,
- Reasonable goal: design, for given r, h and s, an oracle predicting if the reoptimization leads to a better s^{t+1},

[40] T'kindt, V., Raveaux, R. (2022). A learning based matheuristic to solve the two machine flowshop scheduling problem with sum of completion times. *23rd French Conference on Operations Research and Decision Aid (ROADEF)*, Lyon, France.

The ml-VPLS heuristic

- Ideal goal: to have an oracle (predictor) capable of predicting the values of r and h for a given s^t,
- Reasonable goal: design, for given r, h and s, an oracle predicting if the reoptimization leads to a better s^{t+1},
- Use of structured machine learning to solve this classification problem (features based approach, [40]),

[40] T'kindt, V., Raveaux, R. (2022). A learning based matheuristic to solve the two machine flowshop scheduling problem with sum of completion times. *23rd French Conference on Operations Research and Decision Aid (ROADEF)*, Lyon, France.

The ml-VPLS heuristic

- To any $x = [r; h; s]$, we associate a vector $\phi(x) \in \mathbb{R}^{90}$ of 90 features,

The ml-VPLS heuristic

- To any $x = [r; h; s]$, we associate a vector $\phi(x) \in \mathbb{R}^{90}$ of 90 features,
- We want to build a predictor $p(\phi(x), \theta^*) \in [0; 1]$, with $\theta^* \in \Theta$.

The ml-VPLS heuristic

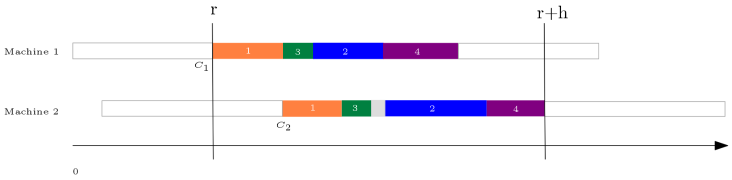
- To any $x = [r; h; s]$, we associate a vector $\phi(x) \in \mathbb{R}^{90}$ of 90 features,
- We want to build a predictor $p(\phi(x), \theta^*) \in [0; 1]$, with $\theta^* \in \Theta$.
  - $\Rightarrow$ When $p(\phi(x), \theta^*) \geq 0.5$, we'll assume that it's worth reoptimizing s in the window $[r; r + h]$.

The `ml-VPLS` heuristic

- To any $x = [r; h; s]$, we associate a vector $\phi(x) \in \mathbb{R}^{90}$ of 90 features,
- We want to build a predictor $p(\phi(x), \theta^*) \in [0; 1]$, with $\theta^* \in \Theta$.
  - $\Rightarrow$ When $p(\phi(x), \theta^*) \geq 0.5$, we'll assume that it's worth reoptimizing s in the window $[r; r + h]$.
- Predictor $p()$ is a neural network and the θ are weights (Deep Learning).

The ml-VPLS heuristic

- A set of 90 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.

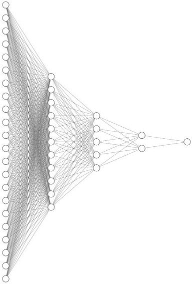
Informative features:

- 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.

The ml-VPLS heuristic

- Predictor (p) is a fully connected neural network:
  - It operates in a vector space ($\in \mathbb{R}^{90}$).
  - Fast inference (prediction time).
  - Other models were put to the test such as 1-dimensional CNNs but inference was too slow.
  - Number of parameters : 14 0000
  - Number of layers : 7
  - Overfitting breakers : Dropout, L1 regularization.



The `ml-VPLS` heuristic: Building the predictor

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

	Train	Validation	Test
#vectors	182 590	184 680	186 086
#1	35.65%	36.19%	36.33%

Table: Data sets description

The `ml-VPLS` heuristic: Building the predictor

- To generate the *training*, *validation* and *test* databases, the same protocol has been used:
  1. Randomly generate 1000 instances of the scheduling problem for a given $n \in \{50; 100\}$,

	Train	Validation	Test
#vectors	182 590	184 680	186 086
#1	35.65%	36.19%	36.33%

Table: Data sets description

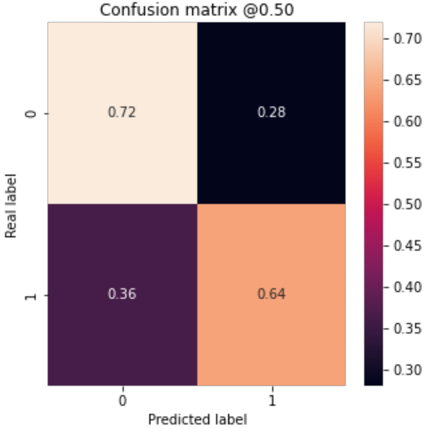
The ml-VPLS heuristic: Building the predictor

- To generate the *training*, *validation* and *test* databases, the same protocol has been used:
  1. Randomly generate 1000 instances of the scheduling problem for a given $n \in \{50; 100\}$,
  2. Run MATH in which all windows $[r; r+h]$ are tested. For each $x = [r; h; s]$ record $\phi(x)$ and the result $y = 1/0$,

	Train	Validation	Test
#vectors	182 590	184 680	186 086
#1	35.65%	36.19%	36.33%

Table: Data sets description

The `ml-VPLS` heuristic: After training



Efficiency of ml-VPLS

- We randomly generate 50 instances per problem size and we compare four versions of the matheuristics:

Efficiency of ml-VPLS

- We randomly generate 50 instances per problem size and we compare four versions of the matheuristics:
  - VPLS: the original matheuristic,

Efficiency of ml-VPLS

- We randomly generate 50 instances per problem size and we compare four versions of the matheuristics:
  - VPLS: the original matheuristic,
  - r-VPLS: random decisions,

Efficiency of `ml-VPLS`

- We randomly generate 50 instances per problem size and we compare four versions of the matheuristics:
  - `VPLS`: the original matheuristic,
  - `r-VPLS`: random decisions,
  - `ml-VPLS`: decisions taken by the predictor,

Efficiency of `ml-VPLS`

- We randomly generate 50 instances per problem size and we compare four versions of the matheuristics:
  - `VPLS`: the original matheuristic,
  - `r-VPLS`: random decisions,
  - `ml-VPLS`: decisions taken by the predictor,
  - `ml-VPLS+`: optimize the 70% best intervals (predictions) at each iteration.

Efficiency of `ml-VPLS`

- We randomly generate 50 instances per problem size and we compare four versions of the matheuristics:
  - `VPLS`: the original matheuristic,
  - `r-VPLS`: random decisions,
  - `ml-VPLS`: decisions taken by the predictor,
  - `ml-VPLS+`: optimize the 70% best intervals (predictions) at each iteration.
- On each instance, `VPLS`, `r-VPLS` and `ml-VPLS` are ran 10 times and the average solution value is used to compute statistics,

Efficiency of ml-VPLS

- We randomly generate 50 instances per problem size and we compare four versions of the matheuristics:
  - VPLS: the original matheuristic,
  - r-VPLS: random decisions,
  - ml-VPLS: decisions taken by the predictor,
  - ml-VPLS+: optimize the 70% best intervals (predictions) at each iteration.
- On each instance, VPLS, r-VPLS and ml-VPLS are ran 10 times and the average solution value is used to compute statistics,
- A total time limit of 60s per instance for VPLS, r-VPLS and ml-VPLS.

Efficiency of ml-VPLS

	$\delta_{avg}(\%)$	$\delta_{max}(\%)$	$T_{avg}(s)$	$T_{max}(s)$	$T2best_{avg}(s)$	$T2best_{max}(s)$
VPLS	0.0031	0.046	61.13	61.36	5.62	22.18
r-VPLS	0.0034	0.060	61.14	61.39	5.88	24.58
ml-VPLS	0.0187	0.083	61.13	61.43	2.55	14.24
ml-VPLS+	0.0055	0.048	7.38	22.87	3.36	15.13

- Results for $n = 50$ jobs -

Efficiency of ml-VPLS

	$\delta_{avg}(\%)$	$\delta_{max}(\%)$	$T_{avg}(s)$	$T_{max}(s)$	$T2best_{avg}(s)$	$T2best_{max}(s)$
VPLS	0.0031	0.046	61.13	61.36	5.62	22.18
r-VPLS	0.0034	0.060	61.14	61.39	5.88	24.58
ml-VPLS	0.0187	0.083	61.13	61.43	2.55	14.24
ml-VPLS+	0.0055	0.048	7.38	22.87	3.36	15.13

- Results for $n = 50$ jobs -

- The trained predictor generalizes well for $n > 50$,

Efficiency of ml-VPLS

	$\delta_{avg}(\%)$	$\delta_{max}(\%)$	$T_{avg}(s)$	$T_{max}(s)$	$T2best_{avg}(s)$	$T2best_{max}(s)$
VPLS	0.0031	0.046	61.13	61.36	5.62	22.18
r-VPLS	0.0034	0.060	61.14	61.39	5.88	24.58
ml-VPLS	0.0187	0.083	61.13	61.43	2.55	14.24
ml-VPLS+	0.0055	0.048	7.38	22.87	3.36	15.13

- Results for $n = 50$ jobs -

- The trained predictor generalizes well for $n > 50$,
- Machine Learning seems interesting to make VPLS converging faster.

Matheuristics: a second pit stop

- In the design of “faster” matheuristics, Machine Learning seems to be an interesting approach,

Matheuristics: a second pit stop

- In the design of “faster” matheuristics, Machine Learning seems to be an interesting approach,
- To drive the search (choice of relevant neighbourhoods to explore),

Matheuristics: a second pit stop

- In the design of “faster” matheuristics, Machine Learning seems to be an interesting approach,
- To drive the search (choice of relevant neighbourhoods to explore),
- Not enough efficient in the above example but improvement is on-going!

Matheuristics: a second pit stop

- In the design of “faster” matheuristics, Machine Learning seems to be an interesting approach,
- To drive the search (choice of relevant neighbourhoods to explore),
- Not enough efficient in the above example but improvement is on-going!
- What is a “good” neighbourhood?

Matheuristics: a second pit stop

- In the design of "faster" matheuristics, Machine Learning seems to be an interesting approach,
- To drive the search (choice of relevant neighbourhoods to explore),
- Not enough efficient in the above example but improvement is on-going!
- What is a "good" neighbourhood?
- We can also imagine other possible use of Machine Learning: selection of variables (set $\mathcal{S}^t$), value of parameters (like k and ℓ in local branching), ...

Outline

Conclusions

- We have seen two examples of matheuristics as *local searches*,

Conclusions

- We have seen two examples of matheuristics as *local searches*,
- A big picture of such approaches,

Dist. based MH	VNS-MH	Var. fixing based MH		
Local Branch.		VPLS	POPMUSIC	Fix & Opt
[9] [14]	[9] [12] [31] [35]	[4] [13] [16] [21] [34]	[18]	[19] [25] [27] [29] [30]

Conclusions

- We have seen two examples of matheuristics as *local searches*,
- A big picture of such approaches,

Dist. based MH	VNS-MH	Var. fixing based MH		
Local Branch.		VPLS	POPMUSIC	Fix & Opt
[9] [14]	[9] [12] [31] [35]	[4] [13] [16] [21] [34]	[18]	[19] [25] [27] [29] [30]

- Matheuristics can be also *constructive heuristics* or can result from the hybridization of *evolutionary algorithms* and MIP....

Constructive MH	Evol. Alg. MH	Others
[14] [17] [23] [25] [26] [27] [32] [35] [38]	[20] [22] [24] [33] [36]	[15] [22] [28] [37] [39]

Conclusions

- (Pro) Matheuristics can be very efficient heuristics for scheduling problems,

Conclusions

- (Pro) Matheuristics can be very efficient heuristics for scheduling problems,
- (Pro) Matheuristics are quite easy to set up,

Conclusions

- (Pro) Matheuristics can be very efficient heuristics for scheduling problems,
- (Pro) Matheuristics are quite easy to set up,
- (Pro) Using non commercial MIP solver is relevant,

Conclusions

- (Pro) Matheuristics can be very efficient heuristics for scheduling problems,
- (Pro) Matheuristics are quite easy to set up,
- (Pro) Using non commercial MIP solver is relevant,
- (Cons) Matheuristics can be time consuming,

Conclusions

- (Pro) Matheuristics can be very efficient heuristics for scheduling problems,
- (Pro) Matheuristics are quite easy to set up,
- (Pro) Using non commercial MIP solver is relevant,
- (Cons) Matheuristics can be time consuming,
- (Cons) Matheuristics can have difficulties to scale up to large size instances (MIP model and CPU time issues),

Conclusions

- (Pro) Matheuristics can be very efficient heuristics for scheduling problems,
- (Pro) Matheuristics are quite easy to set up,
- (Pro) Using non commercial MIP solver is relevant,
- (Cons) Matheuristics can be time consuming,
- (Cons) Matheuristics can have difficulties to scale up to large size instances (MIP model and CPU time issues),
- (Cons) A bunch of parameters to tune.

Conclusions

- (Pro) Matheuristics can be very efficient heuristics for scheduling problems,
- (Pro) Matheuristics are quite easy to set up,
- (Pro) Using non commercial MIP solver is relevant,
- (Cons) Matheuristics can be time consuming,
- (Cons) Matheuristics can have difficulties to scale up to large size instances (MIP model and CPU time issues),
- (Cons) A bunch of parameters to tune.

$\Rightarrow$ Recommendation of the day: if you have a MIP, set up a matheuristic!

Thank you for your attention!

Dist. based MH	VNS-MH	Var. fixing based MH		
Local Branch.		VPLS	POPMUSIC	Fix & Opt
[9] [14]	[9] [12] [31] [35]	[4] [13] [16] [21] [34]	[18]	[19] [25] [27] [29] [30]

Constructive MH	Evol. Alg. MH	Others
[14] [17] [23] [25] [26] [27] [32] [35] [38]	[20] [22] [24] [33] [36]	[15] [22] [28] [37] [39]

[4] Della Croce, F., A. Grosso, F. Salassa (2014). A matheuristic approach for the two-machine completion time flow shop problem, *Annals of Operations Research*, 213:67–78.
[9] Yang, F., Roel, L. (2021). Scheduling hybrid flow shops with time windows, *Journal of Heuristics*, 27:133–158.
[12] Della Croce, F., Salassa, F. (2014). A variable neighborhood search based matheuristic for nurse rostering problems, *Annals of Operations Research*, 218:185–199.
[13] Della Croce, F., Salassa, F., T'kindt, V. (2014). A hybrid heuristic approach for single machine scheduling with release times, *Computers and Operations Research*, 45:7–11.
[14] Smet, P., Wauters, T., Mihaylov, M., Vanden Berghe, G. (2104). This shift minimisation personnel task scheduling problem: A new hybrid approach and computational insights, *Omega*, 46:64-73.
[15] Lin, S.-W., Ying, K.-C. (2016). Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics, *Omega*, 64:115-125.
[16]. Deghdak, K., T'kindt, V., Bouquard, J.-L. (2016). Scheduling evacuation operations, *Journal of Scheduling*, 19:467-478.
[17]. Fanjul-Peyro, L., Perea, F. Ruiz, R. (2017). Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources, *European Journal of Operational Research*, 260:482-493.
[18] Doi, T., Nishi, T., Voss, S. (2018). Two-level decomposition-based matheuristic for airline crew rostering problems with fair working time, *European Journal of Operational Research*, 267:428-438.
[19] Lindahl, M., Sorensen, M., Stidsen, T.R. (2018). A fix-and-optimize matheuristic for university timetabling, *Journal of Heuristics*, 24:645:665.
[20] Monch, L., Roob, S. (2018). A matheuristic framework for batch machine scheduling problems with incompatible job families and regular sum objective, *Applied Soft Computing*, 68:835-846.

Thank you for your attention!

[21] Ta, Q.C., Billaut, J.-C., Bouquard, J.-L. (2018). Matheuristic algorithms for minimizing total tardiness in the m-machine flow-shop scheduling problem, *Journal of Intelligent Manufacturing*, 29:617-628.
[22] Woo, Y.-B., Kim, S. (2018). Matheuristic approaches for parallel machine scheduling problem with time-dependent deterioration and multiple rate-modifying activities, *Computers and Operations Research*, 95:97-112.
[23] Meisel, F., Fagerholt, K. (2019). Scheduling two-way ship traffic for the Kiel Canal: Model, extensions and a matheuristic, *Computers and Operations Research*, 106:119-132.
[24] Ozer, E. A., Sarac, T. (2019). MIP models and a matheuristic algorithm for an identical parallel machine scheduling problem under multiple copies of shared resources constraints, *TOP*, 27:94-124.
[25] Guimaraes, L., Klabjan, D., Almada-Lobo, B. (2013). Pricing, Relaxing and fixing under lot sizing and scheduling, *European Journal of Operational Research*, 230:399-411.
[26] Ferreira, D., Morabito, R., Rangel, S. (2009). Solution approaches for the soft drink integrated production lot-sizing and scheduling problem, *European Journal of Operational Research*, 196:697-706.
[27] James, R.J.W, and Almada-Lobo, B. (2011). Single and parallel machine capacitated lotsizing and scheduling: new iterative MIP-based neighborhood search heuristics, *Computers and Operations Research*, 38:1816-1825.
[28] Kang, S., Malik, K., Thomas, L.J. (1999). Lotsizing and Scheduling on Parallel Machines with Sequence-Dependent Setup Costs, *Management Science*, 45(2):131-295.
[29] Goerler, A., Lalla-Ruiz, E., Voss, S. (2020). Late acceptance Hill-Climbing matheuristic for the general lot sizing and scheduling problem with rich constraints, *Algorithms*, 13(138):1-26.
[30] Thiruvady, D., Blum, C., Ernst, A.T. (2020). Solution merging in matheuristics for resource constrained job scheduling, *Algorithms*, 13(256):1-31.
[31] Ahmadian, M.M., and Salehipour, A. (2021). The just-in-time job-shop scheduling problem with distinct due-dates for operations, *Journal of Heuristics*, 27:175-204.
[32] Chandrasekharan, R. C., Smet, P., Wauters, T. (2021). An automatic constructive matheuristic for the shift minimization personnel task scheduling problem, *Journal of Heuristics*, 27:205-227.
[33] Dang, Q.-V., van Diessen, T., Martagan, T., Adan, I. (2021). A matheuristic for parallel machine scheduling with tool replacement, *European Journal of Operational Research*, 291:640-660.
[34] Della Croce, F., Grosso, A., Salassa, F. (2021). Minimizing total completion time in the two-machine no-idle no-wait flow shop problem, *Journal of Heuristics*, 27:159-173.
[35] Dupin, N., Talbi, E.-G. (2021). Matheuristics to optimize refueling and maintenance planning of nuclear power plants, *Journal of Heuristics*, 27:63-105.

Thank you for your attention!

[36] Guzman, E., Andres, B., Poler, R. (2022). Matheuristic algorithms for Job-Shop scheduling problem using a disjunctive mathematical model, *Computers*, 11(1).
[37] Singh, N., Dang, Q.-V., Akcay, A., Adan, I., Martagan, T. (2022). A matheuristic for AGV scheduling with battery constraints, *European Journal of Operational Research*, 298:855-873.
[38] Hong, J., Moon, K., Lee, K., Lee, K., Pinedo, M.L. (2022). An iterated greedy matheuristic for scheduling in steelmaking-continuous casting process, *International Journal of Production Research*, 60(2):623-643.
[39] Tarhan, I., Oguz, C. (2022). A matheuristic for the generalized order acceptance and scheduling problem, *European Journal of Operational Research*, 299(1):87-103.