# The Feasibility Jump: an LP-free Lagrangian MIP heuristic

Bjørnar Luteberget, **Giorgio Sartor**
*SINTEF, Mathematics and Cybernetics*

# MIP 2022 computational competition

# MIP 2022 Computational Competition

*"Participants are invited to create novel general-purpose primal heuristics for mixed-integer linear optimization problems. Participants are encouraged to develop LP-free heuristics, however, everyone is allowed to solve auxiliary convex optimization problems."*

- Most primal heuristics start after solving the LP relaxation:
  - Feasibility pump, RINS, RENS, Local branching, Pivot and shift, rounding heuristics, etc.

# MIP 2022 Computational Competition

*"Participants are invited to create novel general-purpose primal heuristics for mixed-integer linear optimization problems. Participants are encouraged to develop LP-free heuristics, however, everyone is allowed to solve auxiliary convex optimization problems."*

- Most primal heuristics start after solving the LP relaxation:
  - Feasibility pump, RINS, RENS, Local branching, Pivot and shift, rounding heuristics, etc.

- …and solving the LP relaxation can take some time!

| | Gurobi 9.5 root | Gurobi 9.5 first sol default | Gurobi 9.5 first sol feas emph |
|---|---|---|---|
| – highschool1-aigio | 9.17 s | 501 s | 552 s |
| – neos-5052403-cygnet | 31.6 s | 86.8 s | 44.4 s |
| – s250r10 | 55.0 s | 61.4 s | 8.62 s |

# MIP 2022 Computational Competition

*"Participants are invited to create novel general-purpose primal heuristics for mixed-integer linear optimization problems. Participants are encouraged to develop LP-free heuristics, however, everyone is allowed to solve auxiliary convex optimization problems."*

- Most primal heuristics start after solving the LP relaxation:
  - Feasibility pump, RINS, RENS, Local branching, Pivot and shift, rounding heuristics, etc.

- …and solving the LP relaxation can take some time!

| | Gurobi 9.5 root | Gurobi 9.5 first sol default | Gurobi 9.5 first sol feas emph | Feasibility Jump first sol |
|---|---|---|---|---|
| – highschool1-aigio | 9.17 s | 501 s | 552 s | 2.75 s |
| – neos-5052403-cygnet | 31.6 s | 86.8 s | 44.4 s | 2.87 s |
| – s250r10 | 55.0 s | 61.4 s | 8.62 s | 7.92 s |

# Why are heuristics important?

- Feasible solutions can greatly help a MIP solver:
  - Providing primal bounds, which in turn help the pruning of the branching tree
  - As input to more sophisticated heuristics (e.g., RINS)
  - …

# Why are heuristics important?

- Feasible solutions can greatly help a MIP solver:
  - Providing primal bounds, which in turn help the pruning of the branching tree
  - As input to more sophisticated heuristics (e.g., RINS)
  - …

- Can be extremely relevant in real-life applications[1]:

| **Heuristic** | **CPLEX** | **Heuristic + CPLEX** |
|---|---|---|
| Standalone heuristic | Standalone MIP | Solutions from the heuristic are used as "warm starts" |
| 1.0x   < 1 sec | ~ 2.0x   ~ 1 hour | ~ 0.8x   ~ 1 min |

1. Fischetti, M., Sartor, G. and Zanette, A., 2015. MIP-and-refine matheuristic for smart grid energy management. *International Transactions in Operational Research*, *22*(1), pp.49-59.

# Why are heuristics important?

- Feasible solutions can greatly help a MIP solver:
  - Providing primal bounds, which in turn help the pruning of the branching tree
  - As input to more sophisticated heuristics (e.g., RINS)
  - …
- Can be extremely relevant in real-life applications[1]:

| Heuristic | CPLEX | Heuristic + CPLEX |
|---|---|---|
| Standalone heuristic | Standalone MIP | Solutions from the heuristic are used as "warm starts" |
| 1.0x < 1 sec | ~ 2.0x ~ 1 hour | ~ 0.8x ~ 1 min |

- Sometimes all the customers want is a "good" feasible solution as fast as possible

1. Fischetti, M., Sartor, G. and Zanette, A., 2015. MIP-and-refine matheuristic for smart grid energy management. *International Transactions in Operational Research*, *22*(1), pp.49-59.

# Previous work on pre-root MIP heuristics

- Penalty-based metaheuristics[1]
- Propagation-based heuristics[2]
- …and that's it!

1. Lei, Z., Cai, S., Luo, C. and Hoos, H., 2021, July. Efficient Local Search for Pseudo Boolean Optimization. In International Conference on Theory and Applications of Satisfiability Testing (pp. 332-348). Springer, Cham.
2. Berthold, T. and Hendel, G., 2015. Shift-and-propagate. Journal of Heuristics, 21(1), pp.73-106.

# The Feasibility Jump heuristic

- It's a Lagrangian method
  - $\begin{array}{c}\min c^T x \\ s.t.\ Ax \leq b\end{array} \quad \rightarrow \quad \min\ \lambda^T(b - Ax) \coloneqq \min\ L(x, \lambda)$

    1. Solve $\min L(x; \lambda^*)$
    2. Update $\lambda^*$
    3. Repeat

SINTEF

# The Feasibility Jump heuristic

- It's a Lagrangian heuristic method
  - $\begin{array}{c} \min c^T x \\ s.t.\ Ax \leq b \end{array} \rightarrow \min\ \lambda^T(b - Ax) \coloneqq \min\ L(x, \lambda)$

1. Find a local minimum to $L(x; \lambda^*)$
   in an adaptive discretized neighborhood
1. Update $\lambda^*$
2. Repeat

# The Feasibility Jump heuristic

- It's a Lagrangian heuristic method
  - $\begin{array}{c}\min c^T x \\ s.t.\ Ax \le b\end{array} \quad \rightarrow \quad \min\ \lambda^T(b - Ax) \coloneqq \min\ L(x, \lambda)$

1. Find a local minimum to $L(x; \lambda^*)$
   in an adaptive discretized neighborhood
1. Update $\lambda^*$
2. Repeat

**Three levels of approximation:**

1. Change the value of **one** variable at a time
2. Each variable can "jump" only to a **single** new value
3. The neighborhood defined by 1. and 2. is updated lazily

# The Feasibility Jump heuristic

- It's a Lagrangian heuristic method
  - $\begin{array}{c} \min c^T x \\ s.t.\ Ax \leq b \end{array} \rightarrow \min\ \lambda^T(b - Ax) \coloneqq \min\ L(x, \lambda)$

1. Find a local minimum to $L(x; \lambda^*)$ in an adaptive discretized neighborhood
1. Update $\lambda^*$
2. Repeat

**Three levels of approximation:**

1. Change the value of **one** variable at a time
2. Each variable can "jump" only to a **single** new value
3. The neighborhood defined by 1. and 2. is updated lazily

<u>Up to 1 million iterations per second!</u>

# The Jump value and its score

- The Jump value
  - At the time it is computed, this is the value, different from the incumbent, that maximizes its score

# The Jump value and its score

- The Jump value
  - At the time it is computed, this is the value, different from the incumbent, that maximizes its score

- The score
  - The decrease in the total weighted constraint violation: $\max\{\lambda^T(b - Ax), 0\} - \max\{\lambda^T(b - Ax^*), 0\}$

Total violation before

Total violation after

# The Jump value and its score

- The Jump value
  - At the time it is computed, this is the value, different from the incumbent, that maximizes its score

- The score
  - The decrease in the total weighted constraint violation: $\max\{\lambda^T(b - Ax), 0\} - \max\{\lambda^T(b - Ax^*), 0\}$

Total violation before

Total violation after

- Example
  - $x_1 + x_2 = 5,\ \ x_1, x_2 \in \{1,2,3\},\ \ x_1^*, x_2^* = 0.$
    1. Jump values and scores: $(v_1\!:3, s_1\!:3), (v_2\!:3, s_2\!:3)$. We choose $x_1$. New incumbent: $x_1^* = 3, x_2^* = 0$.

# The Jump value and its score

- The Jump value
  - At the time it is computed, this is the value, different from the incumbent, that maximizes its score

- The score
  - The decrease in the total weighted constraint violation: $\max\{\lambda^T(b - Ax), 0\} - \max\{\lambda^T(b - Ax^*), 0\}$

Total violation before

Total violation after

- Example
  - $x_1 + x_2 = 5,\ \ x_1, x_2 \in \{1,2,3\},\ \ x_1^*, x_2^* = 0.$
    1. Jump values and scores: $(v_1\!: 3, s_1\!: 3), (v_2\!: 3, s_2\!: 3)$. We choose $x_1$. New incumbent: $x_1^* = 3, x_2^* = 0.$
    2. Jump values and scores: $(v_1\!: 2, s_1\!: -1), (v_2\!: 3, s_2\!: 1)$. We choose $x_2$. New incumbent: $x_1^* = 3, x_2^* = 3.$

# The Jump value and its score

- The Jump value
  - At the time it is computed, this is the value, different from the incumbent, that maximizes its score

- The score
  - The decrease in the total weighted constraint violation: $\underbrace{\max\{\lambda^T(b-Ax),0\}}_{\text{Total violation before}} - \underbrace{\max\{\lambda^T(b-Ax^*),0\}}_{\text{Total violation after}}$

- Example
  - $x_1 + x_2 = 5,\ \ x_1, x_2 \in \{1,2,3\},\ \ x_1^*, x_2^* = 0.$
    1. Jump values and scores: $(v_1{:}\,3, s_1{:}\,3), (v_2{:}\,3, s_2{:}\,3)$. We choose $x_1$. New incumbent: $x_1^* = 3, x_2^* = 0.$
    2. Jump values and scores: $(v_1{:}\,2, s_1{:}\,-1), (v_2{:}\,3, s_2{:}\,1)$. We choose $x_2$. New incumbent: $x_1^* = 3, x_2^* = 3.$
    3. Jump values and scores: $(v_1{:}\,2, s_1{:}\,1), (v_2{:}\,2, s_2{:}\,1)$. We choose $x_2$. New incumbent: $x_1^* = 3, x_2^* = 2.$

# The Jump value: example

Consider the problem:

$$x_1 + x_2 = 2,$$
$$x_2 + x_3 \geq 3,$$
$$x_1, x_2, x_3 \in \{0,1,2,3,4\}$$

And the (infeasible) incumbent solution:

$$x_1^* = 0, x_2^* = 0, x_3^* = 0$$

# The Jump value: example

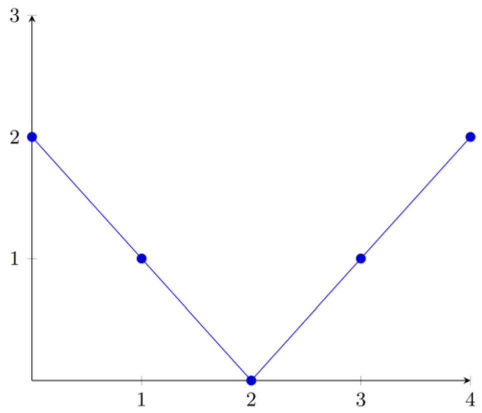
Consider the problem:

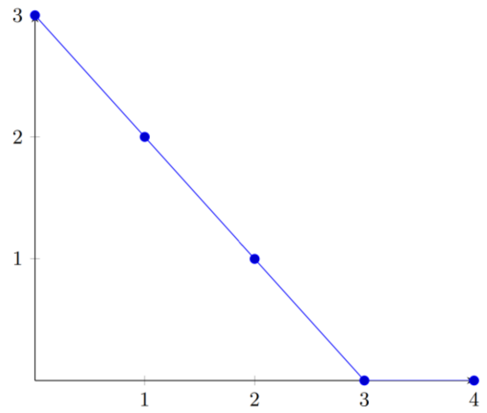
$$x_1 + x_2 = 2,$$
$$x_2 + x_3 \geq 3,$$
$$x_1, x_2, x_3 \in \{0,1,2,3,4\}$$

And the (infeasible) incumbent solution:

$$x_1^* = 0, x_2^* = 0, x_3^* = 0$$

Constraint violation functions for variable $x_2$



# The Jump value: example

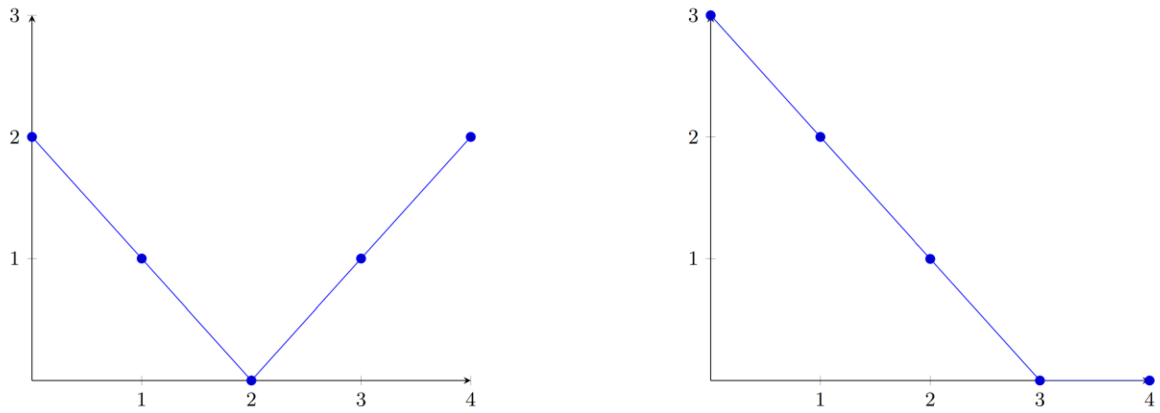
Consider the problem:

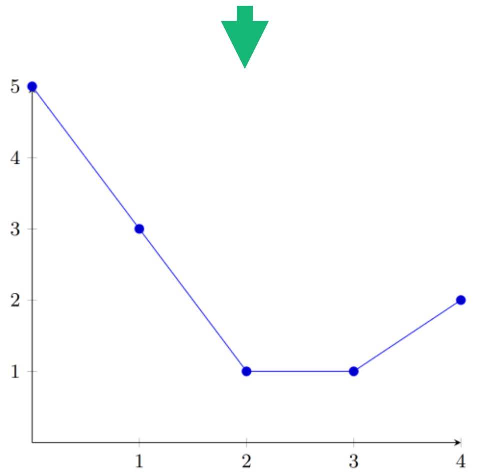
$$x_1 + x_2 = 2,$$
$$x_2 + x_3 \geq 3,$$
$$x_1, x_2, x_3 \in \{0,1,2,3,4\}$$

And the (infeasible) incumbent solution:

$$x_1^* = 0, x_2^* = 0, x_3^* = 0$$

Constraint violation functions for variable $x_2$

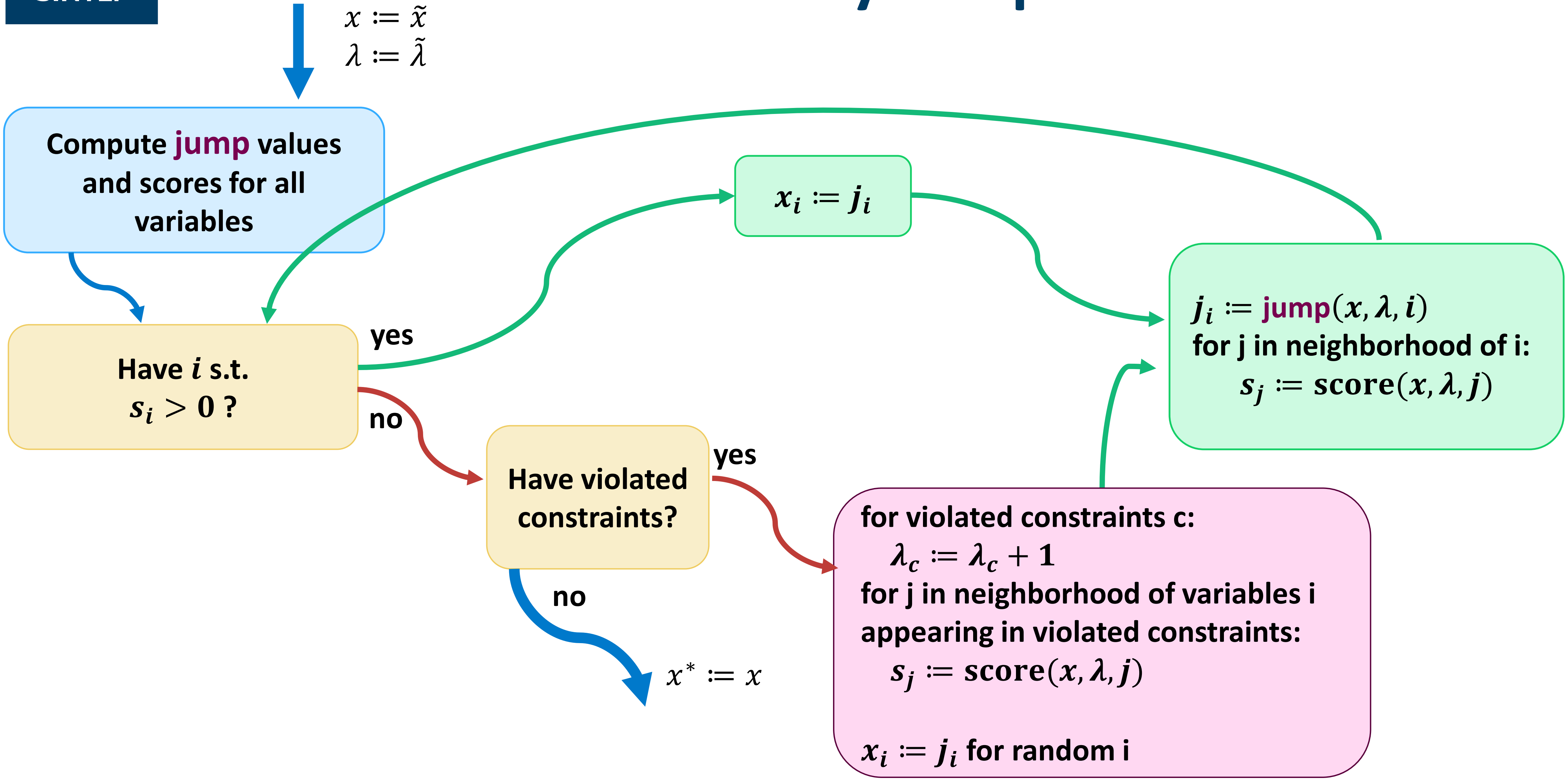




$$\lambda_1, \lambda_2 = 1$$

$$s_2 = 5 - 1 = 4$$

# The Feasibility Jump heuristic



# The Feasibility Jump heuristic

1. Initialize Lagrangian multipliers $\lambda$ to 1
2. Initialize solution $x$
3. Assign to each variable a jump value and compute its corresponding score
4. While $x$ infeasible or termination criterion has not been reached
5. If $\exists$ a variable with positive score
6. Make improving move
7. Compute new jump value and score for this variable
8. Recompute the scores for its neighboring variables
9. Else
10. For every unsatisfied constraint $i$
11. $\lambda_i \leftarrow \lambda_i + 1$
12. Recompute the scores of all variables in the violated constraints
13. Make a random move

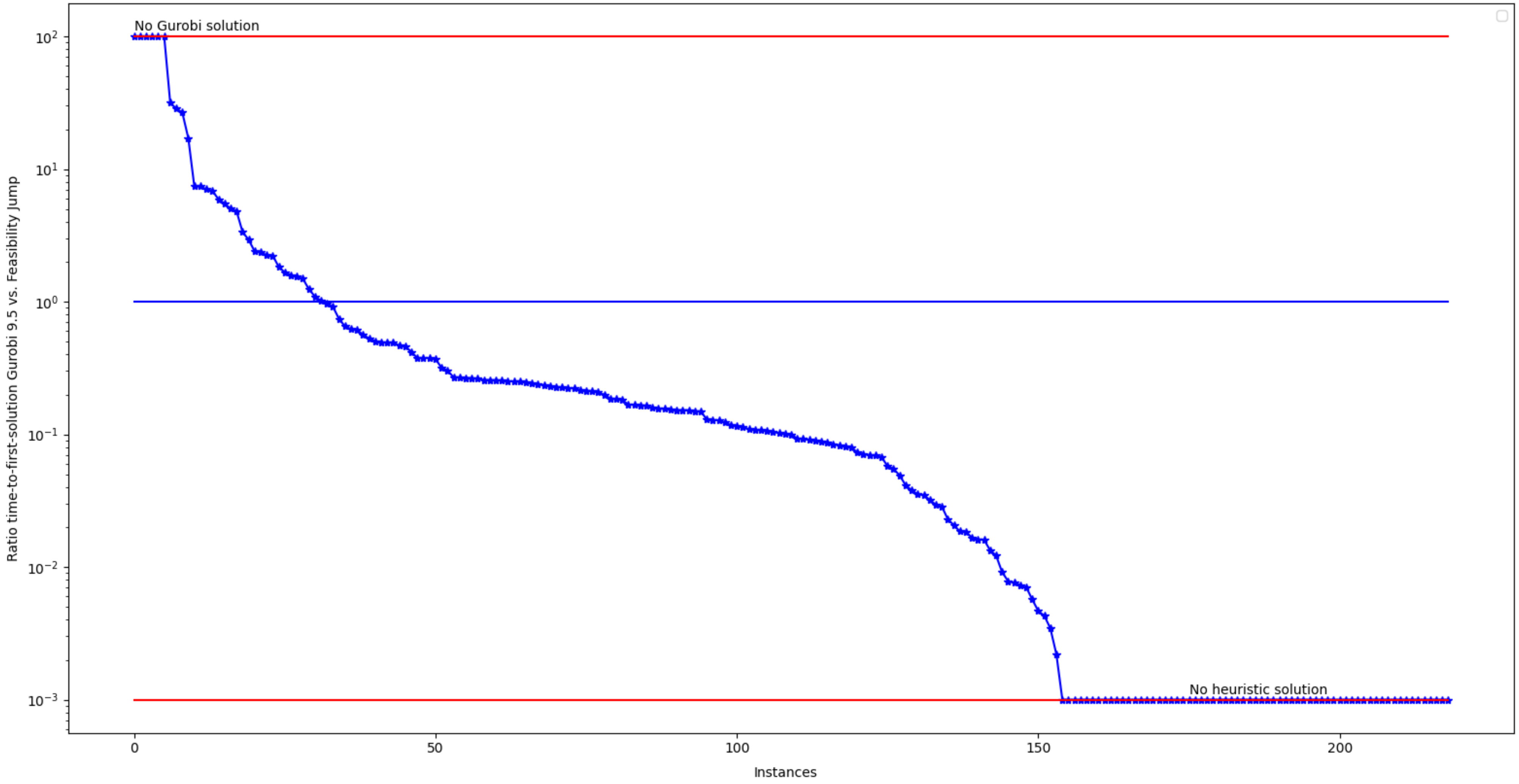
# The Feasibility Jump heuristic

1. Initialize Lagrangian multipliers $\lambda$ to 1
2. Initialize solution $x$
3. Assign to each variable a jump value and compute its corresponding score
4. While $x$ infeasible or termination criterion has not been reached — Constant time
5. If ∃ a variable with positive score — Constant time
6. Make improving move — Constant time
7. Compute new jump value and score for this variable — Proportional to number of constraints in which this variable appears
8. Recompute the scores for its neighboring variables — Proportional to number of nonzeros of the constraints in which this variable appears
9. Else
10. For every unsatisfied constraint $i$ — Proportional to number of violated constraints
11. $\lambda_i \leftarrow \lambda_i + 1$
12. Recompute the scores of all variables in the violated constraints — Proportional to number of nonzeros in the violated constraints
13. Make a random move — Constant time

# MIPLIB 2017 benchmark instances

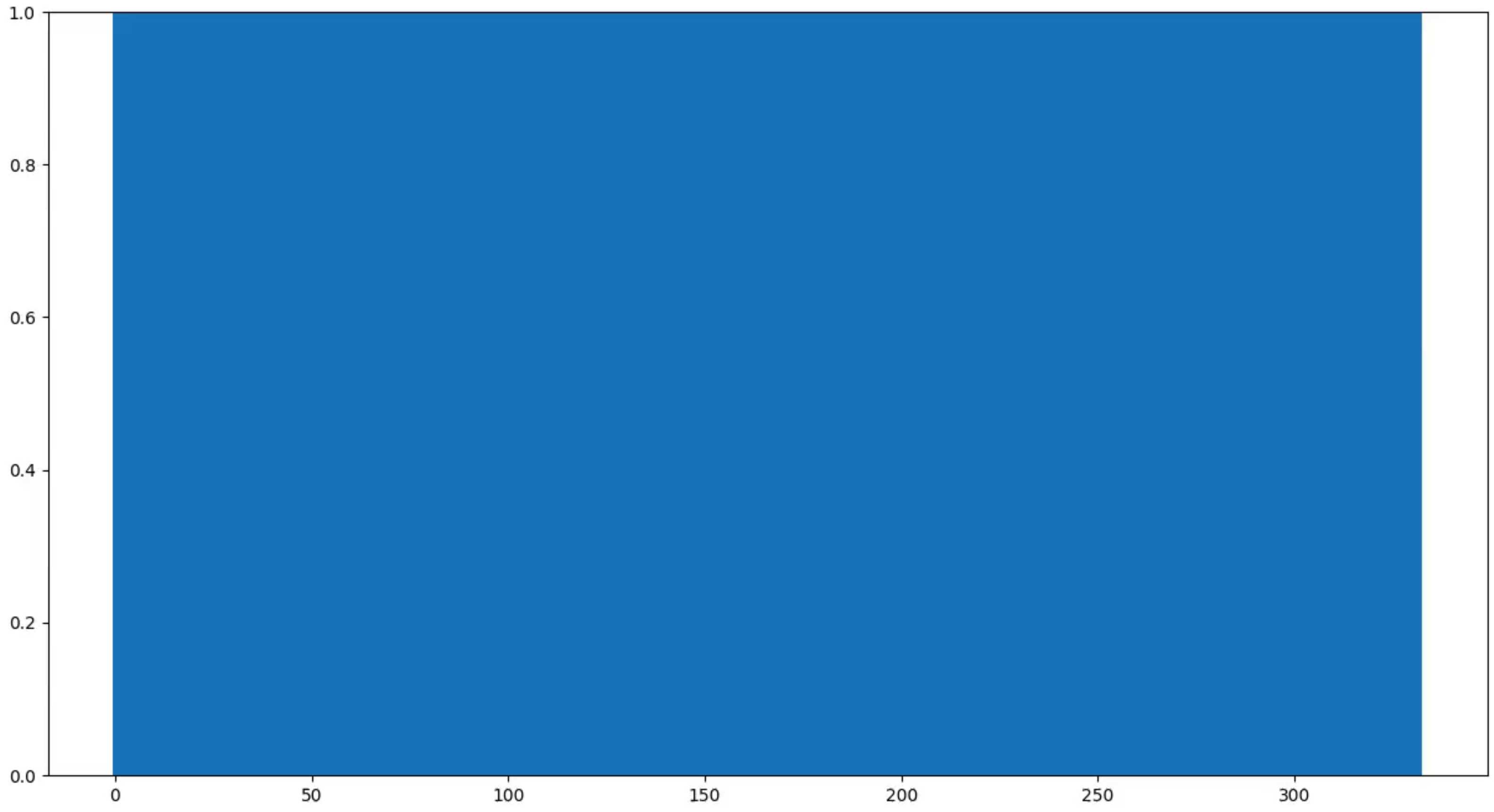
- 240 real-world MIP instances commonly used to compare solvers
- With 1 minute timeout, can we find any feasible solutions?
  - Gurobi 9.5: **213/240**
  - Feasibility Jump (competition version): **154/240**

# Time to first solution: best instances




| | |
|---|---|
| inf | neos-5052403-cygnet.mps |
| inf | neos-4532248-waihi.mps |
| inf | highschool1-aigio.mps |
| inf | s250r10.mps |
| inf | academictimetablesmall.mps |
| inf | physiciansched3-3.mps |
| 31.63 | neos-1354092.mps |
| 26.36 | neos-3004026-krka.mps |
| 17.21 | rmatr200-p5.mps |
| 11.32 | nursesched-medium-hint03.mps |
| 7.03 | neos-5195221-niemur.mps |
| 7.01 | splice1k1.mps |
| 6.89 | ns1760995.mps |
| 6.17 | k1mushroom.mps |
| 5.18 | neos-3555904-turama.mps |
| 4.27 | neos-3381206-awhea.mps |
| 3.32 | dano3_5.mps |
| 3.30 | neos-1456979.mps |
| 2.40 | dano3_3.mps |
| 2.31 | blp-ar98.mps |
| 2.18 | neos-4763324-toguru.mps |
| 2.04 | Istanbul-no-cutoff.mps |
| 1.93 | sing326.mps |
| 1.78 | ns1830653.mps |
| 1.76 | sing44.mps |
| 1.59 | wachplan.mps |
| 1.47 | neos-1582420.mps |
| 1.45 | supportcase7.mps |
| 1.25 | hypothyroid-k1.mps |
| 1.24 | supportcase10.mps |

# Lagrangian multipliers updates: video



# Impact on MIP solvers

- Reference implementation available at https://github.com/sintef/feasibilityjump
- We heard from several implementers of comprehensive optimization solvers
- They reported performance gains from adding Feasibility Jump to their toolbox of heuristics

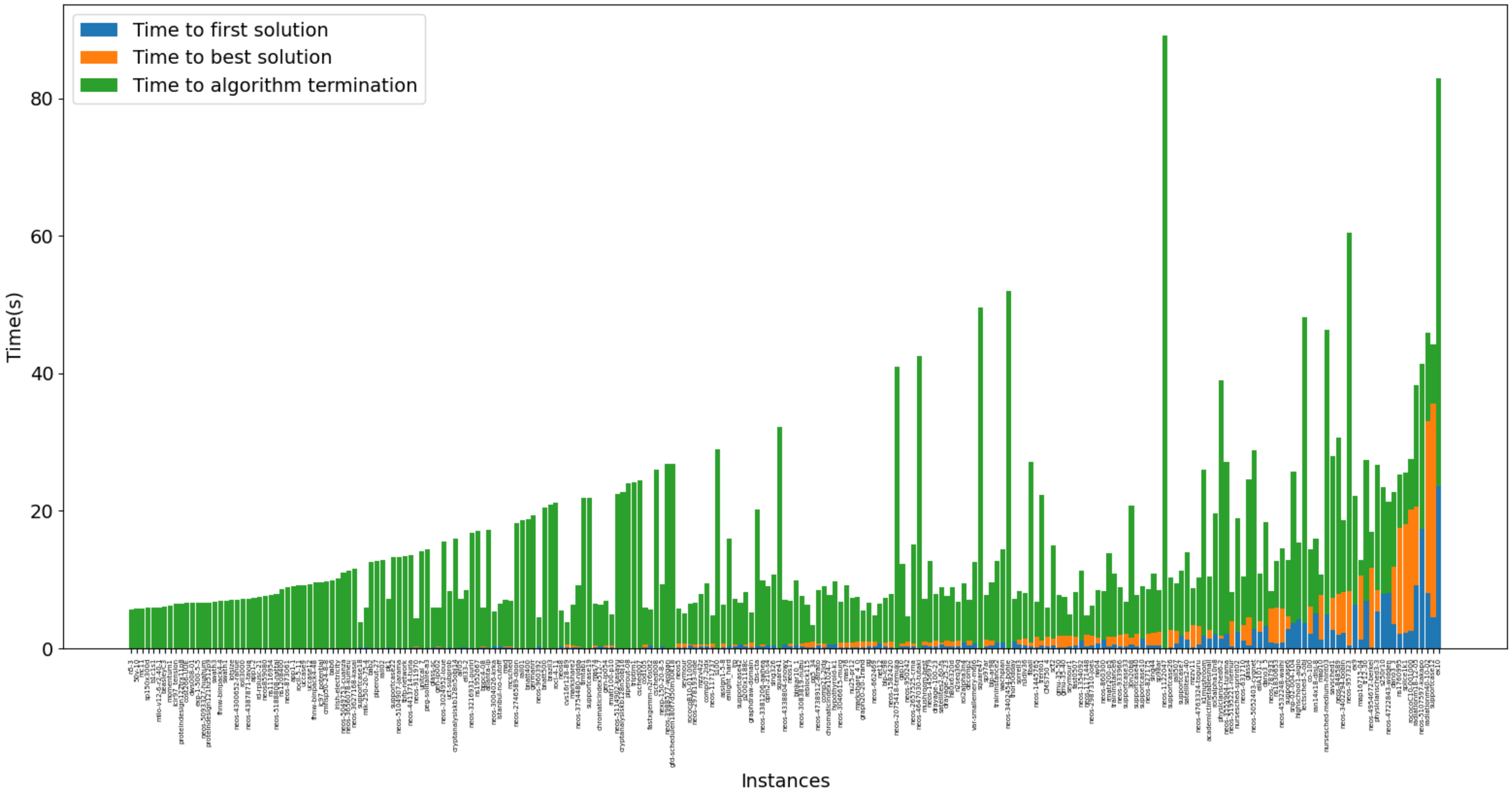
  - FICO Xpress
    - Example: time to first feasible solution of `academictimetablesmall.mps`
    - In version 8.14: 248.0 s
    - In version 9.2: 4.2 s
  - Google OR-tools
  - COPT Cardinal Optimizer
  - HiGHS

# What’s next?

- When does it work and why?
- Machine learning:
  - Better scoring function?
  - Better weighting function?
  - Parameter tuning?
- Local search improvements
  - Additional values other than the jump value?
  - Other memory-based mechanism?
  - Multiple-variable neighborhood?
- When do we terminate?
- Structure-based presolve
- Feasibility Pump + Jump

Technology for a better society

# Behavior till termination



# Results on the MIP competition test instances

| Instance | First sol. time (s) | Last sol. time (s) | Termination time (s) | First sol. objective | Last sol. objective |
|---|---|---|---|---|---|
| academictimetablesmall.mps | 1.07 | 1.2 | 15.39 | 10120 | 3586 |
| comp07-2idx.mps | 0.04 | 0.04 | 10.03 | 1569 | 1569 |
| cryptanalysiskb128n5obj16.mps | - | - | 22.11 | - | - |
| eil33-2.mps | 0.02 | 0.02 | 9.33 | 1642.57 | 1642.57 |
| highschool1-aigio.mps | 2.75 | 2.88 | 12.45 | 14924 | 12956 |
| mcsched.mps | 0.14 | 0.28 | 6.88 | 476402 | 474281 |
| neos-1354092.mps | 0.9 | 0.9 | 10.94 | 400591 | 400591 |
| neos-3024952-loue.mps | - | - | 17.30 | - | - |
| neos-3555904-turama.mps | 2.83 | 2.83 | 27.78 | -34.7 | -34.7 |
| neos-4532248-waihi.mps | 0.95 | 9.31 | 13.94 | 643.2 | 581.76 |
| neos-4722843-widden.mps | - | - | 19.35 | - | - |
| ns1760995.mps | 1.86 | 21.97 | 28.82 | -140.45 | -150.81 |
| ns1952667.mps | - | - | 14.59 | - | - |
| peg-solitaire-a3.mps | - | - | 14.24 | - | - |
| qap10.mps | 0.22 | 0.25 | 12.69 | 534 | 440 |
| rail01.mps | - | - | 19.20 | - | - |
| rococoC10-001000.mps | 21.27 | 21.27 | 27.77 | 42727 | 42727 |
| seymour.mps | 0.01 | 0.29 | 5.27 | 491 | 475 |
| supportcase10.mps | 1.05 | 1.05 | 8.78 | 17 | 17 |

# Some details on the implementation