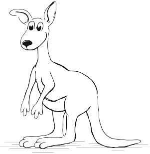


Adaptive Algorithmic Behavior for solving Mixed Integer Programs using Bandit Algorithms

Gregor Hendel Matthias Miltenberger Jakob Witzig

3rd IMI-ISM-ZIB MODAL Workshop
Tokyo,Japan,2018/09/27

Overview

Introduction

Adaptive Large Neighborhood Search

Adaptive LP Pricing

Adaptive Diving

Introduction

Mixed Integer Programs

$$
\begin{aligned}
\min \quad & c^T x \\
\text{s.t.} \quad & Ax \geq b \\
& \ell \leq x \leq u \\
& x \in \{0,1\}^{n_b} \times \mathbb{Z}^{n_i - n_b} \times \mathbb{Q}^{n - n_i}
\end{aligned}
\tag{MIP}
$$

Solution method:

- typically solved with branch-and-cut
- at each node, an LP relaxation is (re-)solved with the dual Simplex algorithm
- primal heuristics, e.g., Large Neighborhood Search and diving methods, support the solution process

- Discrete time steps $t = 1, 2, \dots$
- Finite set of actions $\mathcal{H}$

1. Choose $h_t \in \mathcal{H}$
2. Observe reward $r(h_t, t) \in [0, 1]$
3. Goal: Maximize $\sum_t r(h_t, t)$

- Discrete time steps $t = 1, 2, \ldots$
- Finite set of actions $\mathcal{H}$

1. Choose $h_t \in \mathcal{H}$
2. Observe reward $r(h_t, t) \in [0, 1]$
3. Goal: Maximize $\sum_t r(h_t, t)$

Two main scenarios:

- stochastic i.i.d. rewards for each action over time

The Multi-Armed Bandit Problem

- Discrete time steps $t = 1, 2, \ldots$
- Finite set of actions $\mathcal{H}$

1. Choose $h_t \in \mathcal{H}$
2. Observe reward $r(h_t, t) \in [0, 1]$
3. Goal: Maximize $\sum_t r(h_t, t)$

Two main scenarios:

- stochastic i.i.d. rewards for each action over time
- adversarial an opponent tries to maximize the player's regret.

Literature: [Bubeck and Cesa-Bianchi, 2012]

Bandit Algorithms

Let $T_h(t) = \sum_{t' \leq t} \mathbb{1}_{h=h_t}$ and $\bar{r}_h(t) = \frac{1}{T_h(t)} \sum_{t' \leq t} r_{h,t} \mathbb{1}_{h=h_t}$

ε-greedy

Select heuristic at random with probability $\varepsilon_t = \varepsilon\sqrt{\frac{|\mathcal{H}|}{t}}$, otherwise use best.

Let $T_h(t) = \sum_{t' \leq t} \mathbb{1}_{h=h_t}$ and $\bar{r}_h(t) = \frac{1}{T_h(t)} \sum_{t' \leq t} r_{h,t} \mathbb{1}_{h=h_t}$

ε-greedy

Select heuristic at random with probability $\varepsilon_t = \varepsilon\sqrt{\frac{|\mathcal{H}|}{t}}$, otherwise use best.

Upper Confidence Bound (UCB)

$$h_t \in \begin{cases} \underset{h \in \mathcal{H}}{\operatorname{argmax}} \left\{ \bar{r}_h(t-1) + \sqrt{\frac{\alpha \ln(1+t)}{T_h(t-1)}} \right\} & \text{if } t > |\mathcal{H}|, \\ \{H_t\} & \text{if } t \leq |\mathcal{H}|. \end{cases}$$

Let $T_h(t) = \sum_{t' \leq t} \mathbb{1}_{h=h_t}$ and $\bar{r}_h(t) = \frac{1}{T_h(t)} \sum_{t' \leq t} r_{h,t} \mathbb{1}_{h=h_t}$

ε-greedy

Select heuristic at random with probability $\varepsilon_t = \varepsilon \sqrt{\frac{|\mathcal{H}|}{t}}$, otherwise use best.

Upper Confidence Bound (UCB)

$$h_t \in \begin{cases} \operatorname*{argmax}_{h \in \mathcal{H}} \left\{ \bar{r}_h(t-1) + \sqrt{\frac{\alpha \ln(1+t)}{T_h(t-1)}} \right\} & \text{if } t > |\mathcal{H}|, \\ \{H_t\} & \text{if } t \leq |\mathcal{H}|. \end{cases}$$

Exp.3

$$p_{h,t} = (1-\gamma) \cdot \frac{\exp(w_{h,t})}{\sum_{h'} \exp(w_{h',t})} + \gamma \cdot \frac{1}{|\mathcal{H}|}$$

Let $T_h(t) = \sum_{t' \leq t} \mathbb{1}_{h=h_t}$ and $\bar{r}_h(t) = \frac{1}{T_h(t)} \sum_{t' \leq t} r_{h,t} \mathbb{1}_{h=h_t}$

ε-greedy

Select heuristic at random with probability $\varepsilon_t = \varepsilon \sqrt{\frac{|\mathcal{H}|}{t}}$, otherwise use best.

Upper Confidence Bound (UCB)

$$h_t \in \begin{cases} \operatorname*{argmax}_{h \in \mathcal{H}} \left\{ \bar{r}_h(t-1) + \sqrt{\frac{\alpha \ln(1+t)}{T_h(t-1)}} \right\} & \text{if } t > |\mathcal{H}|, \\ \{H_t\} & \text{if } t \leq |\mathcal{H}|. \end{cases}$$

Exp.3

$$p_{h,t} = (1-\gamma) \cdot \frac{\exp(w_{h,t})}{\sum_{h'} \exp(w_{h',t})} + \gamma \cdot \frac{1}{|\mathcal{H}|}$$

Individual parameters $\alpha, \varepsilon, \gamma \geq 0$ can be calibrated to the problem at hand.

Adaptive Large Neighborhood Search

Auxiliary MIP

Let P be a MIP with solution set $\mathcal{F}_P$. For a polyhedron $\mathcal{N} \subseteq \mathbb{Q}^n$ and objective coefficients $c_{\text{aux}} \in \mathbb{Q}^n$, a MIP P^{aux} defined as

$$\min \left\{ c_{\text{aux}}^T x \,|\, x \in \mathcal{F}_P \cap \mathcal{N} \right\}$$

is called an auxiliary MIP of P, and $\mathcal{N}$ is called neighborhood.

Large Neighborhood Search (LNS) heuristics solve auxiliary MIPs and can be distinguished by their respective neighborhoods.

Famous LNS Heuristics

- Relaxation Induced Neighborhood Search (RINS) [Danna et al., 2005]
- Local Branching [Fischetti and Lodi, 2003]
- Crossover, Mutation [Rothberg, 2007]
- RENS [Berthold, 2014]
- Proximity [Fischetti and Monaci, 2014]
- DINS [Ghosh, 2007]
- Zeroobjective [in SCIP, Gurobi, XPress,...]
- Analytic Center Search [Berthold et al., 2017]
- ...

Goal A suitable reward function $r^{\text{alns}}(h_t, t) \in [0, 1]$

Rewarding Neighborhoods

Goal A suitable reward function $r^{\text{alns}}(h_t, t) \in [0, 1]$

Solution Reward

$$r^{\text{sol}}(h_t, t) = \begin{cases} 1 & , \text{if } x^{\text{old}} \neq x^{\text{new}} \\ 0 & , \text{else} \end{cases}$$

Goal A suitable reward function $r^{\text{alns}}(h_t, t) \in [0, 1]$

Solution Reward

$$r^{\text{sol}}(h_t, t) = \begin{cases} 1 & \text{, if } x^{\text{old}} \neq x^{\text{new}} \\ 0 & \text{, else} \end{cases}$$

Gap Reward

$$r^{\text{gap}}(h_t, t) = \frac{c^T x^{\text{old}} - c^T x^{\text{new}}}{c^T x^{\text{old}} - c^{\text{dual}}}$$

Goal A suitable reward function $r^{\text{alns}}(h_t, t) \in [0, 1]$

Solution Reward

$$r^{\text{sol}}(h_t, t) = \begin{cases} 1 & \text{, if } x^{\text{old}} \neq x^{\text{new}} \\ 0 & \text{, else} \end{cases}$$

Gap Reward

$$r^{\text{gap}}(h_t, t) = \frac{c^T x^{\text{old}} - c^T x^{\text{new}}}{c^T x^{\text{old}} - c^{\text{dual}}}$$

Failure Penalty

$$r^{\text{fail}}(h_t, t) = \begin{cases} 1, & \text{if } x^{\text{old}} \neq x^{\text{new}} \\ 1 - \phi(h_t, t)\frac{n(h_t)}{n^{\text{lim}}} & \end{cases}$$

Goal A suitable reward function $r^{\text{alns}}(h_t, t) \in [0, 1]$

Solution Reward

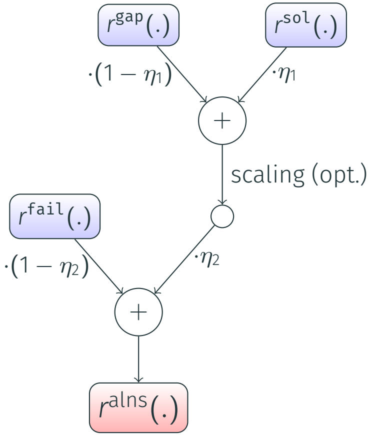
$$r^{\text{sol}}(h_t, t) = \begin{cases} 1 & \text{, if } x^{\text{old}} \neq x^{\text{new}} \\ 0 & \text{, else} \end{cases}$$

Gap Reward

$$r^{\text{gap}}(h_t, t) = \frac{c^T x^{\text{old}} - c^T x^{\text{new}}}{c^T x^{\text{old}} - c^{\text{dual}}}$$

Failure Penalty

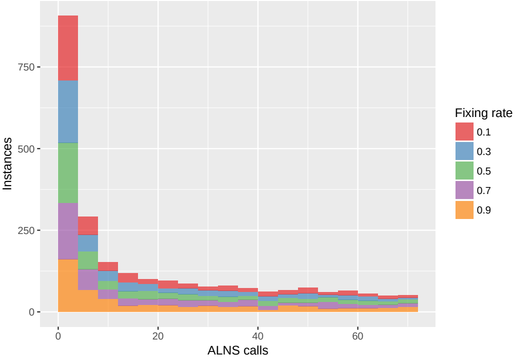
$$r^{\text{fail}}(h_t, t) = \begin{cases} 1, & \text{if } x^{\text{old}} \neq x^{\text{new}} \\ 1 - \phi(h_t, t) \frac{n(h_t)}{n^{\text{lim}}} & \end{cases}$$



Default settings in ALNS: $\eta_1 = 0.8$, $\eta_2 = 0.5$

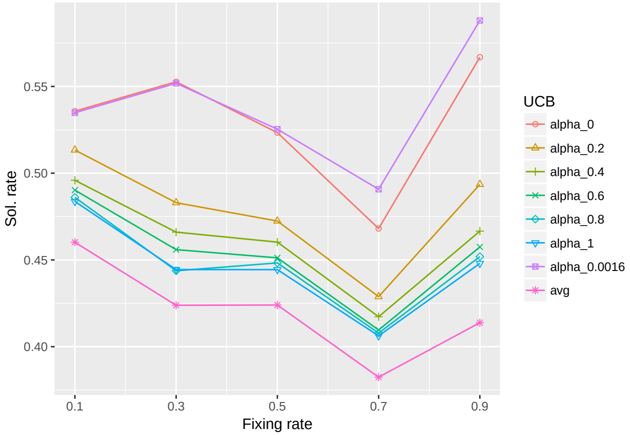
Simulation for parameter calibration

- Always execute all 8 neighborhoods with ALNS (disable old LNS heuristics)
- Disable solution transfer
- Record each reward
- Fixing rates 0.1 – 0.9



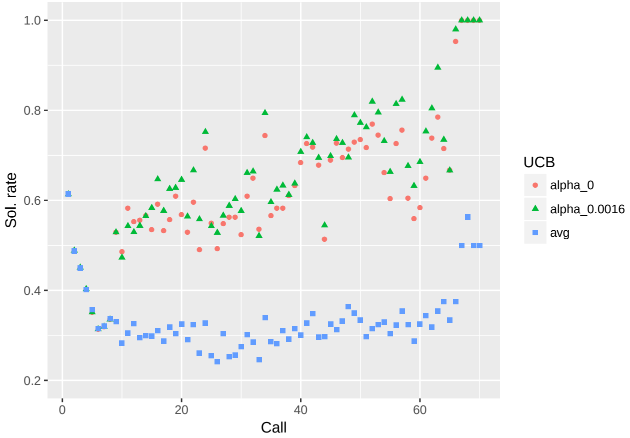
Test Set
666 instances from the test sets MIPLIB3, MIPLIB2003, MIPLIB2010, Cor@l, 5h time limit.

Simulate 100 repetitions of UCB, Exp.3, and ϵ-greedy on the data



$$h_t \in \begin{cases} \underset{h \in \mathcal{H}}{\operatorname{argmax}} \left\{ \bar{r}_h(t-1) + \sqrt{\frac{\alpha \ln(1+t)}{T_h(t-1)}} \right\} & \text{if } t > |\mathcal{H}|, \\ \{H_t\} & \text{if } t \leq |\mathcal{H}|. \end{cases} \tag{UCB}$$

Learning Curve of UCB

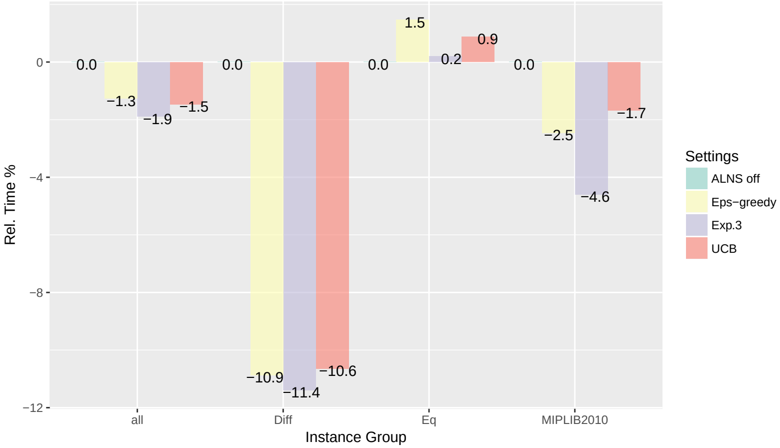


$$h_t \in \begin{cases} \underset{h \in \mathcal{H}}{\operatorname{argmax}} \left\{ \bar{r}_h(t-1) + \sqrt{\frac{\alpha \ln(1+t)}{T_h(t-1)}} \right\} & \text{if } t > |\mathcal{H}|, \\ \{H_t\} & \text{if } t \leq |\mathcal{H}|. \end{cases} \tag{UCB}$$

Adaptive Large Neighborhood Search

- new primal heuristic plugin `heur_alns.c`
- controls 8 neighborhoods
- neighborhoods are called based on their reward
- further algorithmic steps: generic fixings, adaptive fixing rate
- released with SCIP 5.0

Performance of the ALNS framework



Adaptive LP Pricing

LP Pricing Selection

SCIP features the parameter `lp/pricing = ...`

	`s(teepest edge)` [Forrest and Goldfarb, 1992]	`d(evex)` [Harris, 1973]	`q(uick start steep)`
`neos-1601936`	1098.50	2126.55	1502.57
`nw04`	46.90	21.34	31.08
`pigeon-12`	3600.00	3600.00	3.02

Automatic selection strategy within SoPlex: run devex for 10000 iterations, then switch to steepest edge.

LP Pricing Selection

SCIP features the parameter `lp/pricing = ...`

	`s(teepest edge)` [Forrest and Goldfarb, 1992]	`d(evex)` [Harris, 1973]	`q(uick start steep)`
`neos-1601936`	1098.50	2126.55	1502.57
`nw04`	46.90	21.34	31.08
`pigeon-12`	3600.00	3600.00	3.02

Automatic selection strategy within SoPlex: run devex for 10000 iterations, then switch to steepest edge.

Goal: Maximize LP throughput

LP Pricing goal and setup

Maximize LP throughput $\Leftrightarrow$ discover and select the LP pricing with minimum expected running time τ_p^*, $p \in \{\text{devex}, \text{steep}, \text{qsteep}\}$

Problem for UCB: Need $[0, 1]$ score to maximize

LP Pricing goal and setup

Maximize LP throughput $\Leftrightarrow$ discover and select the LP pricing with minimum expected running time τ_p^*, $p \in \{\text{devex}, \text{steep}, \text{qsteep}\}$

Problem for UCB: Need $[0, 1]$ score to maximize

Solution: Scale the (normalized) reward

- Let $\tau_{t,p}$ be the measured running time for pricer p at step t
- Use reward $r_{t,p} = \frac{1}{1+\frac{\tau_{t,p}}{\bar{\tau}_p}}$ for UCB

LP Pricing goal and setup

Maximize LP throughput $\Leftrightarrow$ discover and select the LP pricing with minimum expected running time τ_p^*, $p \in \{\text{devex}, \text{steep}, \text{qsteep}\}$

Problem for UCB: Need $[0, 1]$ score to maximize

Solution: Scale the (normalized) reward

- Let $\tau_{t,p}$ be the measured running time for pricer p at step t
- Use reward $r_{t,p} = \frac{1}{1+\frac{\tau_{t,p}}{\bar{\tau}_p}}$ for UCB

1st alternative: UCB variant (shifted greedy) (thanks to Tobias Achterberg)

- select a favorite pricer, w.l.o.g. p_1
- use shift vector $\sigma \in {\mathbb{R}_+}^{\mathcal{P}}$ $\sigma_{p_1} = 100$, $\sigma_p = 50$ for $p \neq p_1$
- always start with p_1 for a couple of resolves
- only start selection process if average iterations of p_1 exceed a threshold, e.g., 20.
- always select the pricer that minimizes

$$\bar{\tau}_p^\sigma = \frac{\sum_t \mathbb{1}_{p_t=p}\tau_{t,p}}{T_p(t-1) + \sigma_p}$$

LP Pricing goal and setup

Maximize LP throughput $\Leftrightarrow$ discover and select the LP pricing with minimum expected running time τ_p^*, $p \in \{\text{devex}, \text{steep}, \text{qsteep}\}$

Problem for UCB: Need $[0, 1]$ score to maximize

Solution: Scale the (normalized) reward

- Let $\tau_{t,p}$ be the measured running time for pricer p at step t
- Use reward $r_{t,p} = \frac{1}{1+\frac{\tau_{t,p}}{\bar{\tau}_p}}$ for UCB

1st alternative: UCB variant (shifted greedy) (thanks to Tobias Achterberg)

- select a favorite pricer, w.l.o.g. p_1
- use shift vector $\sigma \in \mathbb{R}_+{}^{\mathcal{P}}$ $\sigma_{p_1} = 100$, $\sigma_p = 50$ for $p \neq p_1$
- always start with p_1 for a couple of resolves
- only start selection process if average iterations of p_1 exceed a threshold, e.g., 20.
- always select the pricer that minimizes

$$\bar{\tau}_p^\sigma = \frac{\sum_t \mathbb{1}_{p_t=p}\tau_{t,p}}{T_p(t-1) + \sigma_p}$$

2nd alternative: Turn shifted greedy weights into weighted sampling weights

- compute shifted version of average as in shifted greedy
- sample from weight distribution $w_{p,t} \propto \frac{1}{\bar{\tau}_p^\sigma + 10^{-4}}$

Computational Results

LP Solver CPLEX 12.7.1

Group	#	Pricing	solved	LP throughput abs.	LP throughput rel.	Time abs.	Time rel.
all	593	devex	288	72.4	1.000	152.30	1.000
		qsteep	289	74.7	1.032	144.93	0.952
		steep	288	76.4	1.056	147.34	0.967
		weighted	289	73.0	1.009	148.40	0.974
		UCB	292	79.6	1.100	147.56	0.969
		sh. greedy	292	80.8	1.117	143.94	0.945

LP Solver SoPlex 3.1.1

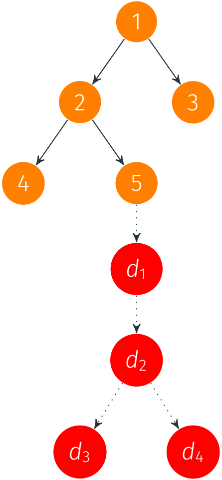
Group	#	Pricing	solved	LP throughput abs.	LP throughput rel.	Time abs.	Time rel.
all	587	devex	279	44.2	1.000	167.36	1.000
		qsteep	272	35.0	0.793	181.74	1.086
		steep	280	37.7	0.854	178.01	1.064
		weighted	282	42.7	0.966	170.75	1.020
		UCB	284	45.5	1.031	168.82	1.009
		sh. greedy	288	50.5	1.144	163.93	0.980

Test set: 150 instances from a total of 666 (MIPLIBs & Cor@l), time limit, default + 3 LP Seeds, 48 node cluster with 16 Intel Xeon Gold 5122 @ 3.60GHz, 96GB, Ubuntu 16.04

Adaptive Diving

Diving Heuristics

9 different diving heuristics explore an auxiliary tree in probing mode.



Diving Heuristics in SCIP [Achterberg, 2007]

- coefficient diving
- fractionality diving
- guided diving [Danna et al., 2005]
- pseudo costs
- ...

Information from Diving:

- Primal solutions
- Variable branching history (pseudo costs, ...)
- Conflict clauses

Goal of Selection
Improving both primal solutions and relevant search information

Problem: Solutions are only rarely found by diving heuristics, see also [Khalil et al., 2017].

Possible reward measures that discriminate better:

- minimum avg. depth
- minimum backtracks/conflict ratio
- minimum avg. probing nodes
- minimum avg. LP iterations

Unlikely that there is a unique best diving algorithm $\Rightarrow$ use weighted sampling method with inverse probabilities as in LP pricing.

Computational Results

Group	#	Setting	Solved	Time	rel.
all	1477	default	1005	152.54	1.000
		adaptive diving	1020	146.05	0.957
$\geq$ 100 sec.	396	default	363	485.39	1.000
		adaptive diving	378	436.99	0.900

Setup:

adaptive diving selects from 9 diving heuristics. It is called in addition to the SCIP diving heuristics.

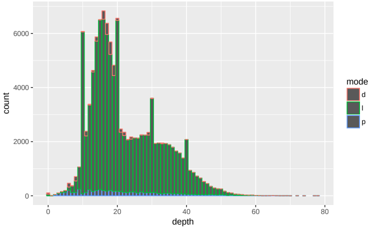
Test set: 496 instances from MIPLIBs & Cor@l benchmark sets 1h time limit, default + 2 LP Seeds, 48 node cluster with 16 Intel Xeon Gold 5122 @ 3.60GHz, 96GB, Ubuntu 16.04

Instance,seed pairs are treated as individual observations.

- bandit selection variants for LP pricing selection, diving heuristics, and Large Neighborhood Search heuristics
- different scenarios require different reward functions and selection strategies
- adaptive selection yields computational benefits in all three cases.

In the future, we would like to

- finalize the LP pricing prototype
  - switch to deterministic LP time measurement
  - calibrate bandit parameters
  - exploit seemingly lognormal distribution of LP solving time for simulation and different bandit algorithm (Thompson sampling)
- investigate the usefulness of keeping learned information for future solves.

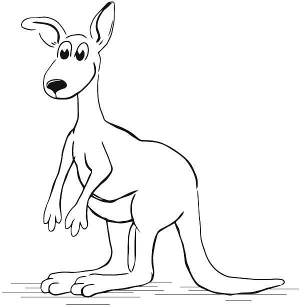


LP counts in diving, probing, and normal lp mode for `timtab1`.

More on our work

- Gregor Hendel, Matthias Miltenberger, and Jakob Witzig, *Adaptive Algorithmic Behavior for Solving Mixed Integer Programs Using Bandit Algorithms*, ZIB-Report 18-36, Zuse Institute Berlin, 2018
- Gregor Hendel, *Adaptive Large Neighborhood Search for MIP*, under preparation.

Thank you for your attention!



Visit scip.zib.de.

Achterberg, T. (2007).
Constraint Integer Programming.
PhD thesis, Technische Universität Berlin.

Berthold, T. (2014).
Rens - the optimal rounding.
Mathematical Programming Computation, 6(1):33–54.

Berthold, T., Perregaard, M., and Meszaros, C. (2017).
Four good reasons to use an interior point solver within a mip solver.

Bubeck, S. and Cesa-Bianchi, N. (2012).
Regret analysis of stochastic and nonstochastic multi-armed bandit problems.
CoRR, abs/1204.5721.

Danna, E., Rothberg, E., and Pape, C. L. (2005).
Exploring relaxation induced neighborhoods to improve MIP solutions.
Mathematical Programming, 102(1):71–90.

Bibliography ii

Fischetti, M. and Lodi, A. (2003).
Local branching.
Mathematical Programming, 98(1-3):23–47.

Fischetti, M. and Monaci, M. (2014).
Proximity search for 0-1 mixed-integer convex programming.
Technical report, DEI - Università di Padova.

Forrest, J. J. and Goldfarb, D. (1992).
Steepest-edge simplex algorithms for linear programming.
Math. Program., 57:341–374.

Ghosh, S. (2007).
DINS, a MIP Improvement Heuristic.
In Fischetti, M. and Williamson, D. P., editors, *Integer Programming and Combinatorial Optimization: 12th International IPCO Conference, Ithaca, NY, USA, June 25-27, 2007. Proceedings*, pages 310–323, Berlin, Heidelberg. Springer Berlin Heidelberg.

Harris, P. M. J. (1973).
Pivot selection methods of the devex lp code.
Mathematical Programming, 5(1):1–28.

Khalil, E. B., Dilkina, B., Nemhauser, G. L., Ahmed, S., and Shao, Y. (2017).
Learning to run heuristics in tree search.
In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 659–666. AAAI Press.

Rothberg, E. (2007).
An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions.
INFORMS Journal on Computing, 19(4):534–541.