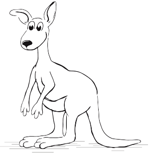


Introducing Adaptive Algorithmic Behavior of Primal Heuristics in SCIP for Solving Mixed Integer Programs

Gregor Hendel, hendel@zib.de
joint work with Matthias Miltenberger and Jakob Witzig

Monash University, Melbourne, Australia, 22 March, 2019

Zuse Institute Berlin – Fast Algorithms, Fast Computers

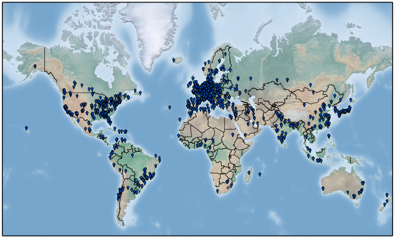
A research institute and computing center of the State of Berlin with research units:

- Numerical Analysis and Modeling
- Visualization and Data Analysis
- Optimization:
  Energy – Transportation – Health – Mathematical Optimization Methods
- Scientific Information Systems
- Computer Science and High Performance Computing

Meet the SCIP Team

26 active developers

- 4 running Bachelor and Master projects
- 14 running PhD projects
- 8 postdocs and professors



4 development centers in Germany

- ZIB: SCIP, SoPlex, UG, ZIMPL
- TU Darmstadt: SCIP and SCIP-SDP
- FAU Erlangen-Nürnberg: SCIP
- RWTH Aachen: GCG

Many international contributors and users

- more than 10 000 downloads per year from 100+ countries

Careers

- 10 awards for Masters and PhD theses: MOS, EURO, GOR, DMV
- 7 former developers are now building commercial optimization software at CPLEX, FICO Xpress, Gurobi, MOSEK, and GAMS

Overview

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

Introduction

Large Neighborhood Search for MIP

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.

Typical LNS Neighborhoods

Let $\mathcal{M} \subseteq \{1, \dots, n_i\}$, $x^* \in \mathbb{Q}^n$.

- Fixing neighborhood

$$\mathcal{N}^{\text{fix}}(\mathcal{M}, x^*) := \{x \in \mathbb{Q}^n \mid x_j = x_j^* \; \forall j \in \mathcal{M}\}$$

Typical LNS Neighborhoods

Let $\mathcal{M} \subseteq \{1, \ldots, n_i\}$, $x^* \in \mathbb{Q}^n$.

- Fixing neighborhood

$$\mathcal{N}^{\text{fix}}(\mathcal{M}, x^*) := \{x \in \mathbb{Q}^n \mid x_j = x_j^* \; \forall j \in \mathcal{M}\}$$

- Improvement neighborhood

$$\mathcal{N}^{\text{obj}}(\delta, x^{\text{inc}}) := \left\{x \in \mathbb{Q}^n \mid c^T x \leq (1 - \delta) \cdot c^T x^{\text{inc}} + \delta \cdot c^{\text{dual}}\right\}$$

Examples of LNS Heuristics

Relaxation Induced Neighborhood Search (RINS) [Danna et al., 2005]

$$\mathcal{N}_{\text{RINS}} := \mathcal{N}^{\text{fix}}\left(\mathcal{M}^{=}\left(\left\{x^{\text{lp}}, x^{\text{inc}}\right\}\right), x^{\text{inc}}\right) \cap \mathcal{N}^{\text{obj}}\left(\delta, x^{\text{inc}}\right).$$

Relaxation Induced Neighborhood Search (RINS) [Danna et al., 2005]

$$\mathcal{N}_{\text{RINS}} := \mathcal{N}^{\text{fix}}\left(\mathcal{M}^{=}\left(\left\{x^{\text{lp}}, x^{\text{inc}}\right\}\right), x^{\text{inc}}\right) \cap \mathcal{N}^{\text{obj}}\left(\delta, x^{\text{inc}}\right).$$

Local Branching [Fischetti and Lodi, 2003]

$$\mathcal{N}_{\text{LBranch}} := \left\{x \in \mathbb{Q}^{n} \mid \left\|x - x^{\text{inc}}\right\|_{b} \leq d_{\max}\right\} \cap \mathcal{N}^{\text{obj}}(\delta, x^{\text{inc}})$$

Famous LNS Heuristics

- Relaxation Induced Neighborhood Search (RINS) [Danna et al., 2005]
- Local Branching [Fischetti and Lodi, 2003]

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]
- ...

Introduction

Multi-Armed Bandit Selection

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 $\sum_t r(h_t, t)$

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

- 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]

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)

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

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)

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

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.

SCIP's Adaptive LNS

Adaptive Large Neighborhood Search

- new primal heuristic plugin `heur_alns.c`
- controls 8 neighborhoods
- neighborhoods are bandit-selected based on their reward
- further algorithmic steps: generic fixings, adaptive fixing rate
- released with SCIP 5.0, improved in SCIP 6.0

SCIP's Adaptive LNS

Reward Function for LNS

Rewarding Neighborhoods

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

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

Rewarding Neighborhoods

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

Solution Reward

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

Gap Reward

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

Rewarding Neighborhoods

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

Solution Reward

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

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

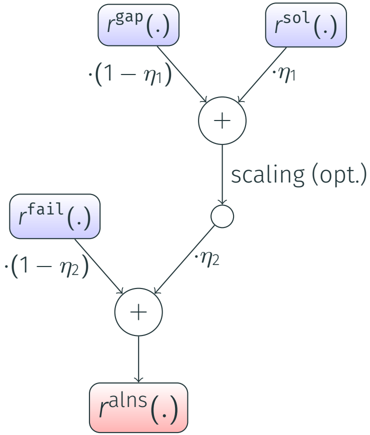
$$r^{\text{sol}}(h_t, t) = \begin{cases} 1 & \text{, if } x^{\text{old}} \neq x^{\text{new}} \\ 0 & \text{, else} \end{cases}$$

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

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



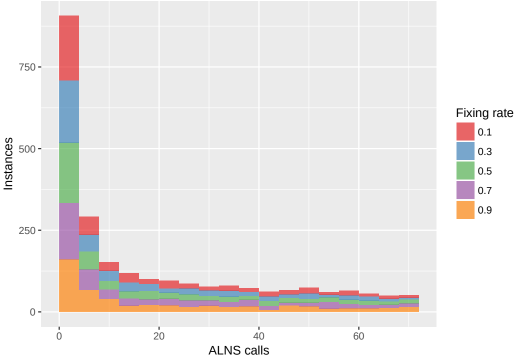
Default settings in ALNS: $\eta_1 = 0.8$, $\eta_2 = 0.5$

SCIP's Adaptive LNS

Computational Results

Simulation for Parameter Calibration

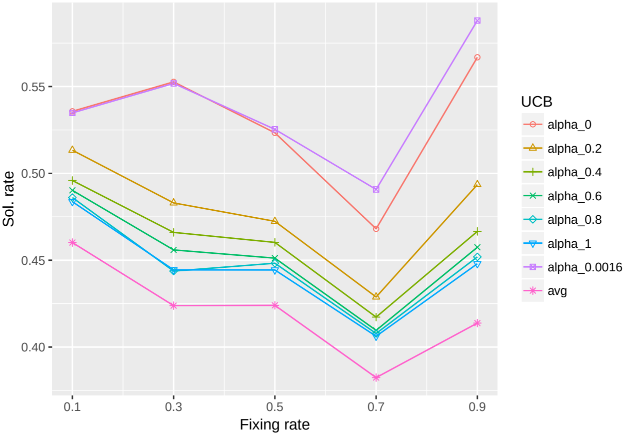
- Always execute all 8 neighborhoods with ALNS (disable old LNS heuristics)
- Disable solution transfer
- Record each reward
- Fixing rates 0.1 – 0.9



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

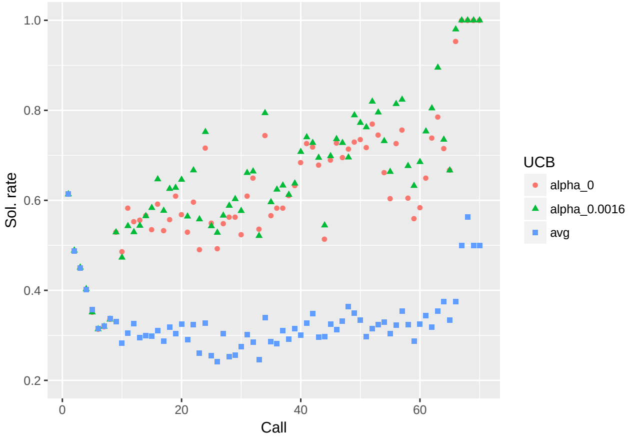
UCB Calibration

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



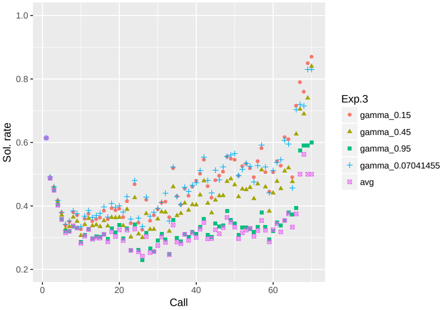
$$h_t \in \begin{cases} \underset{h \in \mathcal{H}}{\operatorname{argmax}} \left\{ \bar{r}_h(t-1) + \sqrt{\frac{\alpha \ln(1+t)}{T_h(t-1)}} \right\} & \text{if } t > |\mathcal{H}|, \\ \{H_t\} & \text{if } t \leq |\mathcal{H}|. \end{cases} \tag{UCB}$$

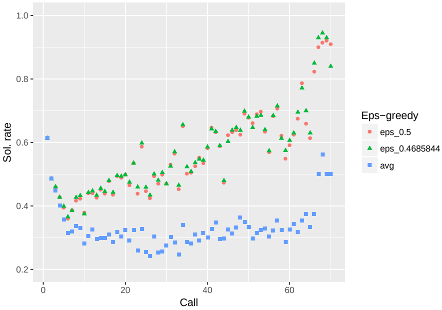
Learning Curve of UCB



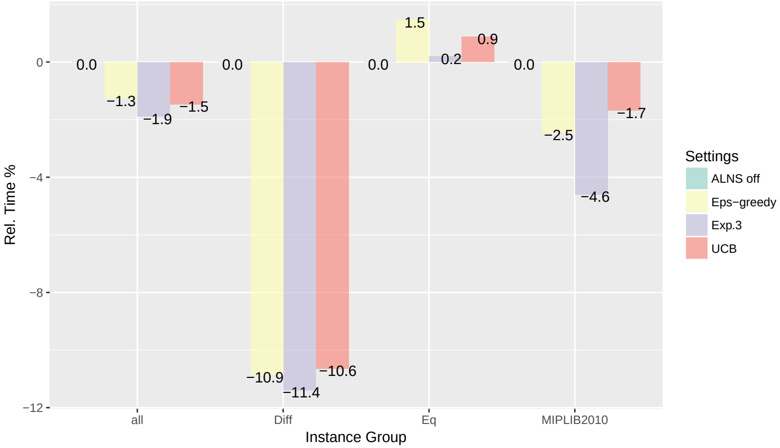
$$h_t \in \begin{cases} \operatorname*{argmax}_{h \in \mathcal{H}} \left\{ \bar{r}_h(t-1) + \sqrt{\frac{\alpha \ln(1+t)}{T_h(t-1)}} \right\} & \text{if } t > |\mathcal{H}|, \\ \{H_t\} & \text{if } t \leq |\mathcal{H}|. \end{cases} \tag{UCB}$$

More Learning Curves



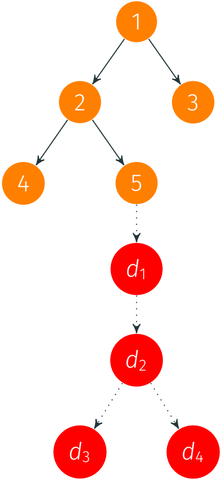


Performance of the ALNS Framework



Diving & Adaptive Diving

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

Reward Functions for Diving

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.

Adaptive Diving

Adaptive Diving will be one of the main new features in SCIP 7.0. It

- automatically incorporates user diving heuristics[1], if the user sets the visibility to `public`.
- Selects via weighted sampling based on conflicts/backtrack, or simply revolves through the available diving strategies.
- Provides new score types, including number of found solutions.
- learns, by default, from its own calls, but also from individual runs of the heuristics.

[1] see SCIP Docu for information how to write diving heuristics.

Outlook: 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

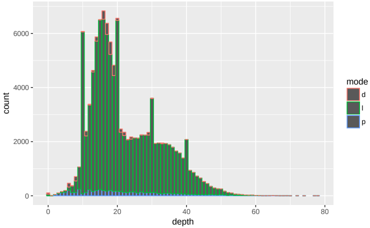
Test set: 150 instances from a total of 666 (MIPLIBs & Cor@l), time limit, default + 3 LP Seeds, 48 node cluster with 16 Intel Xeon Gold 5122 @ 3.60GHz, 96GB, Ubuntu 16.04

Conclusion

- 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.

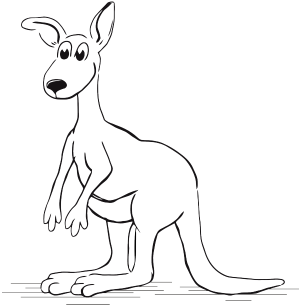


LP counts in diving, probing, and normal lp mode for `timtab1`.

Further Literature

- Gregor Hendel, Matthias Miltenberger, and Jakob Witzig, *Adaptive Algorithmic Behavior for Solving Mixed Integer Programs Using Bandit Algorithms*, OR 2018: International Conference on Operations Research, accepted for publication, Preprint
- Gregor Hendel, *Adaptive Large Neighborhood Search for MIP*, December 2018, under review, Preprint.
- Ambros Gleixner et al., *The SCIP Optimization Suite 6.0*, ZIB-Report 18-26, July 2018, Link

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.

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.