Operations Research Letters 41 (2013) 611–614

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Measuring the impact of primal heuristics

Timo Berthold *

Zuse Institute Berlin, Department of Optimization, Takustr. 7, 14195 Berlin, Germany

ARTICLE INFO

Article history:
Received 27 March 2013
Received in revised form
19 August 2013
Accepted 20 August 2013
Available online 28 August 2013

Keywords:
Mixed integer programming
Primal heuristic
Performance measure
Primal integral

ABSTRACT

In modern MIP solvers, primal heuristics play a key role in finding high-quality solutions. However, classical performance measures reflect the impact of primal heuristics on the overall solving process badly. In this article, we introduce a new performance measure, the "primal integral", which depends on the quality of solutions and on the time when they are found. We compare five state-of-the-art MIP solvers w.r.t. the newly proposed measure, and show that heuristics improve their performance by up to 80%.

1. Introduction

When implementing optimization software, two questions naturally arise: how does the new code perform with respect to existent codes, and which are the best settings for a particular algorithm? This goes back to the early days of operations research: Hoffman et al. reported a first computational experiment to compare different implementations of linear programming algorithms in 1953 [9]. Just as researchers and software vendors want to distinguish their code on general test sets, a user wants to tune an optimization software for a particular set of problems. However, all parties require suitable criteria for measuring the performance of a software implementation.

With the rise of computational research, standards and guidelines for conducting computational experiments were proposed [5,13,11,16]. One key issue of the cited articles is the choice of suitable performance indicators. In mathematical programming, the running time to optimality is the "gold standard" for performance comparisons. For branch-and-bound based algorithms, the number of branch-and-bound nodes is another typical measure. Similarly, when using a simplex or an interior point based solver, the number of iterations is commonly used. Both the number of iterations and the number of nodes attempt to estimate the running time by a measure that is less dependent on the hardware and at the same time better reflects the sheer computational complexity.

The setting. In this article, we will use mixed integer (linear) programming as a showcase for our computational experiments. *Mixed integer programming* (MIP) is to solve the optimization problem

$$\tilde{x}_{\text{opt}} \in \operatorname{argmin}\{c^T x \mid Ax \leq b, x_j \in \mathbb{Z} \text{ for all } j \in J\},$$

with $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $J \subseteq \{1, \ldots, n\}$. Mixed integer programs are a typical example for optimization problems that can be efficiently solved by a branch-and-bound method.

One advantage of branch-and-bound based algorithms, as opposed to, for example, pure cutting plane algorithms, is that suboptimal (though feasible) solutions, so-called *incumbents*, are often available early during the course of the algorithm. *Primal heuristics* boost this characteristic even further. As a consequence, sufficiently good solutions might be available long before the branch-and-bound search terminates. A *primal heuristic* is, roughly speaking, an incomplete algorithm that aims at finding high-quality feasible solutions quickly. In general, it is neither guaranteed to be successful, nor does it provide dual information on the quality of the solution by itself.

Existing measures. The time needed to find a first feasible solution, an optimal solution, or a solution within a certain gap to optimality (see, e.g., [10]) are performance criteria that concentrate on the primal part of the solution process. Each of these has its individual strengths and weaknesses. The time to first solution entirely disregards the solution quality: for about one quarter (23 of 87) of the MIPLIB 2010 [15] benchmark instances, a trivial solution of all variables set to their lower bound (or all to their upper bound) is feasible, but most of the times useless. Particularly when analyzing heuristics embedded in a complete solver, the time to the first solution mainly measures the time needed for preprocessing and solving the root node relaxation: the MIP solvers CPLEX, GUROBI and XPRESS find solutions for 72, 70, and 64 of the 84 feasible MIPLIB 2010 benchmark instances (three instances are infeasible) during

* Tel.: +49 30 84185 425; fax: +49 30 84185 269.
E-mail address: berthold@zib.de.

0167-6377/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.orl.2013.08.007

root node processing. The time to optimal solution, however, ignores that slightly suboptimal but practically sufficient solutions might have been found long before. Finally, taking the time to a certain gap is an attempt to balance this, but the choice of the threshold is arbitrary by design.

Altogether, the important consideration for primal heuristics should be the trade-off between speed and solution quality. None of the above measures entirely meets this requirement. In marked contrast, two of the named measures, time to first solution and time to optimality, rather represent extreme points. It is the goal of this paper to introduce a new performance measure that reflects the development of the solution quality over the complete optimization process.

The challenge. We argue that standard performance criteria are not well suited to describe the impact that primal heuristics have within a solver. Take the following observation. On the one hand, for state-of-the-art MIP solvers, the impact of primal heuristics on the overall running time and the number of branch-and-bound nodes is typically minor to negligible (5–15%, according to recent presentations by software vendors), whereas other components such as cutting planes or branching rules change these numbers by a factor of 2 or 3 (see, e.g., [2,3]).

On the other hand, the solver vendors, such as CPLEX, GUROBI or XPRESS, seem to consider primal heuristics to be a "trade secret". It stays unrevealed which heuristics they use, when they are called, or just how many of them a solver features, whereas, for other components, there are plenty of user parameters and statistical outputs available. One interpretation of this discrepancy might be that primal heuristics are considered a – if not the – crucial part of the software, and their value is simply not reflected by the performance measures that we commonly use.

2. The primal integral

In this article, we introduce a new performance measure, in particular for benchmarking primal heuristics, that takes into account the whole solution process. The goal is to measure the progress of the primal bound's convergence towards the optimal solution over the entire solving time. Therefore, we make use of the *primal gap* of a feasible solution, consider this as a function over time, and compute the integral of that function.

Definition 2.1. Let $\tilde{x}$ be a solution for a MIP, and $\tilde{x}_{\text{opt}}$ be an optimal (or best known) solution for that MIP. We define the *primal gap* $\gamma \in [0, 1]$ of $\tilde{x}$ as follows:

$$\gamma(\tilde{x}) := \begin{cases} 0, & \text{if } \left|c^T\tilde{x}_{\text{opt}}\right| = \left|c^T\tilde{x}\right| = 0, \\ 1, & \text{if } c^T\tilde{x}_{\text{opt}} \cdot c^T\tilde{x} < 0, \\ \dfrac{\left|c^T\tilde{x}_{\text{opt}} - c^T\tilde{x}\right|}{\max\{|c^T\tilde{x}_{\text{opt}}|, |c^T\tilde{x}|\}}, & \text{else.} \end{cases}$$

We assume that $c^T\tilde{x}_{\text{opt}} \leq c^T\tilde{x}$. For a computational evaluation this means that $\tilde{x}_{\text{opt}}$ needs to be "updated" in the case that an improved solution is found for an unsolved instance.

Note that for two feasible MIP solutions $\tilde{x}_1, \tilde{x}_2$ with $c^T\tilde{x}_1 < c^T\tilde{x}_2$ and $\text{sgn}(c^T\tilde{x}_2) = \text{sgn}(c^T\tilde{x}_{\text{opt}})$ it holds that $\gamma(\tilde{x}_1) < \gamma(\tilde{x}_2)$. Now, assume that we have available the objective function values of intermediate incumbent solutions and the points in time when they have been found, for a given MIP solver, a certain problem instance, and a fixed computational environment.

Definition 2.2. Let $t_{\max} \in \mathbb{R}_{\geq 0}$ be a limit on the solution time of a MIP solver. Its *primal gap function* p: $[0, t_{\max}] \mapsto [0, 1]$ is defined as follows:

$$p(t) := \begin{cases} 1, & \text{if no incumbent until point } t, \\ \gamma(\tilde{x}(t)), & \text{with } \tilde{x}(t) \text{ being the incumbent solution} \\ & \text{at point } t, \text{ else.} \end{cases}$$

The primal gap function $p(t)$ is a step function that changes whenever a new incumbent is found. It is monotonically decreasing and 0 from the point at which the optimal solution is found.

Definition 2.3. Let $T \in [0, t_{\max}]$, and let $t_i \in [0, T]$ for $i \in 1, \ldots, I-1$ be the points in time when a new incumbent solution is found, $t_0 = 0, t_I = T$. We define the *primal integral* $P(T)$ of a run as follows:

$$P(T) := \int_{t=0}^{T} p(t)\,dt = \sum_{i=1}^{I} p(t_{i-1}) \cdot (t_i - t_{i-1}).$$

We suggest using $P(t_{\max})$ for measuring the quality of primal heuristics. It features two simple, but important, attributes. First, whenever a better solution is found at the same point in time, $P(t_{\max})$ decreases. Second, whenever the same solution is found at an earlier point in time, $P(t_{\max})$ decreases. Briefly, the primal integral favors finding good solutions early. For the performance measures discussed in the introduction, at most one of these two attributes holds in general.

The fraction $\frac{P(t_{\max})}{t_{\max}}$ can be seen as the average solution quality during the search process. Said differently, the smaller $P(t_{\max})$ is, the better is the expected quality of the incumbent solution if we stop the solver at an arbitrary point in time. The primal integral is an absolute measure in the sense that it is only defined by a single solver, unlike, for instance, a *performance profile* which compares relative performance and is defined by a set of solvers. Precisely speaking, *performance profiles* [6] represent the relative performance of a set of algorithms compared to a *virtual best solver* as a curve in a graph. A performance profile shows what percentage of the instances (which is the ordinate) in a given test set a given solver could solve within a time factor (which is the abscissa) of the best solver.

Note that, for pure feasibility problems (instances with a zero objective function), the primal integral and the overall running time will give the same measure, up to a constant scaling factor. This follows from the simple observation that for feasibility instances, the primal gap is 1 before the solution is found (and the solution process thereby terminates) and 0 afterwards. Hence, when for two different runs the running time differs by a factor k, the primal integral $p(t)$ will also differ by a factor k.

In a recent work on rounding heuristics for MINLP, Nannicini and Belotti [18] used the percentage of total running time for which one given algorithm gave rise to a strictly better solution than another one to compare two solution processes. This can be formulated in terms of our notation as

$$\frac{1}{T}\int_{t=0}^{T} \mathbf{1}_{\{p_1 > p_2\}}(t)\,dt,$$

with p_1 and p_2 the primal gap functions of two runs of a solver (with different settings) and $\mathbf{1}_{\{p_1>p_2\}}(t)$ a characteristic function, being 1 if $p_1(t) > p_2(t)$ and 0 otherwise. The main differences between this measure and the primal integral are that this measure neither takes the actual objective function values of the solutions into account, nor is it an absolute measure, since it compares the relative performance of two algorithms.

3. Computational experiments

We conducted experiments with five state-of-the-art MIP solvers: CBC 2.7.0 [4], FICO XPRESS 23.01.06 [7], GUROBI 5.1.0 [8], IBM ILOG CPLEX 12.5.0 [12], and SCIP 3.0.1 [1], compiled with SOPLEX 1.7.1 [19] as the LP solver. CPLEX, GUROBI, and XPRESS are among the fastest commercial MIP solvers, and CBC and SCIP are among the fastest open-source MIP solvers [17]. Further, these

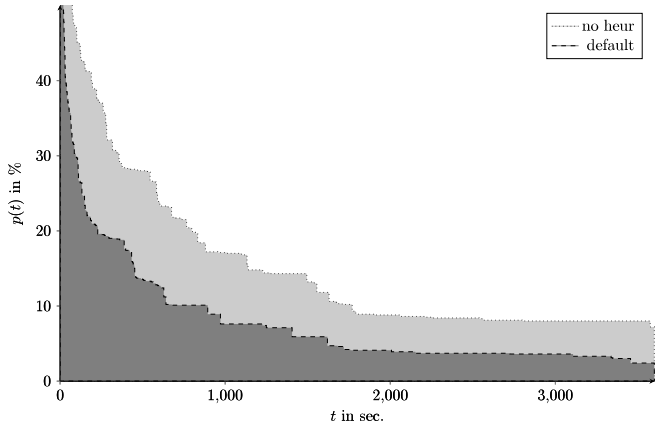

Fig. 1. Course of the primal gap when running SCIP with and without primal heuristics.

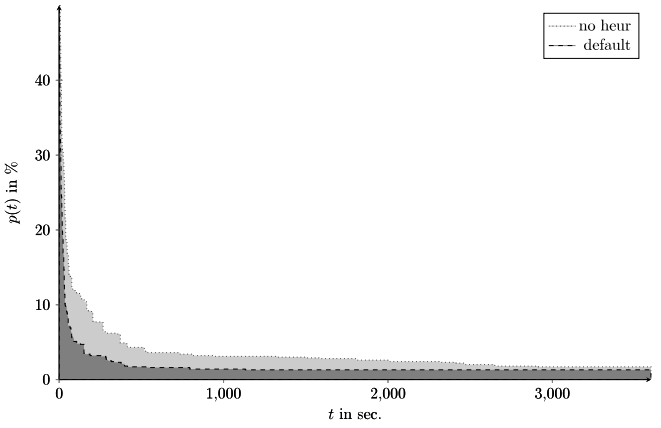

Fig. 2. Course of the primal gap when running CPLEX with and without primal heuristics.

are the five codes that have been used to compile the MIPLIB 2010 [15] benchmark set which we chose as a test set for our computational experiments. We excluded the three infeasible instances `ash608gpia-3col`, `enlight14`, and `ns1766074`. Additionally, we excluded the instance `mspp16`, because SCIP and CBC ran out of memory. This leaves 83 instances in the test set.

The results for running the solvers in default mode are taken from the benchmarks for optimization software webpage of Hans Mittelmann [17], as of 20 February 2013. Additional results with disabled primal heuristics were obtained on the actual same computer, a 64 bit Intel Xeon X5680 CPU at 3.20 GHz with 12 MB cache and 32 GB main memory, running an OPENSUSE 12.1 with a GCC 4.6.2 compiler. Turboboost was disabled. In all experiments, there was only one job at a time to avoid random noise in the measured running time that might be caused by cache misses if multiple processes share common resources.

As a first test, we compare the performance of SCIP and CPLEX when running with and without primal heuristics. We show the evolution of the primal gap in Figs. 1 and 2. The dashed line corresponds to the average (taken over 83 instances) primal gap function, when running the solver in default mode. The dark area corresponds to the average primal integral. Accordingly, the dotted line and the light shaded (plus the dark shaded) area correspond to the average primal gap function and the average primal integral when running the solver without heuristics.

For SCIP, the average value of $\frac{P(t_{\max})}{t_{\max}}$ was 9.05% when using primal heuristics and 16.18% without. For CPLEX, it was 1.92% and 3.58%, respectively. This indicates that, on this test set, for these two solvers, primal heuristics lead to an improvement of 78.8% and 86.5% in the primal bound, on average. As a comparison, the running time to optimality was 819.0 s with and 910.5 s without primal heuristics for SCIP (+11%); 181.1 s and 239.9 s for CPLEX (+32%). Both solvers solved four instances fewer when not using primal heuristics. Note that for both solvers the "default" function is strictly smaller than the "no heur" function. This implies that, independent of the chosen time limit, using primal heuristics is superior in terms of the primal integral.

Of course, we can use the primal integral as another metric to compare the performance of solvers against each other. It has been argued in [15] (and previously in [13]) that more than one performance indicator should be used for evaluating mathematical programming software, ideally based on different aspects of the optimization process. Fig. 3 shows the course of the average primal gap function for CBC, SCIP, XPRESS, GUROBI, CPLEX, and a *virtual best solver* (VBS). VBS takes for each instance the minimum of the primal bounds of the five solvers at each point in time. The results for the average primal integral, in particular the order of the solvers, are clearly different from those for the mean time

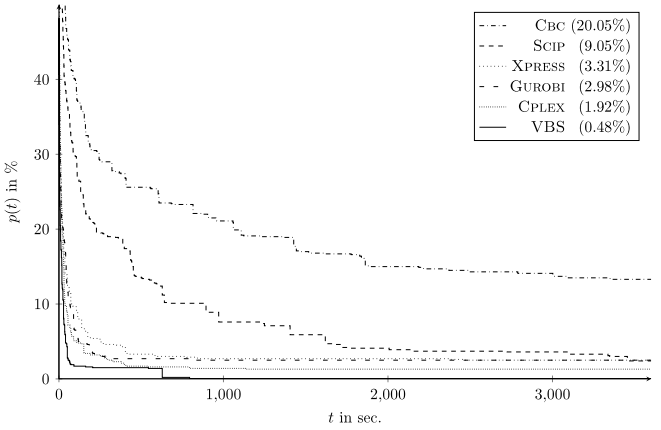

Fig. 3. Course of the primal gap for five different MIP solvers plus a virtual best solver.

to optimality at [17], which shows that different solvers have different strengths.

Even more important, averages tell you little about a single instance. The primal integral of the virtual best solver in Fig. 3 is a factor 3.8 smaller than the primal integral of the best individual solver, meaning that the portfolio of solvers is significantly better than any single solver. For 82 of the 83 instances, at least one solver found an optimal solution within 800 s. In contrast, for each individual solver, there are at least four instances for which the solver did not find an optimal solution after one hour. Each solver contributes to VBS, meaning that each solver has the single best primal bound for some instances for some time. Altogether, this shows once again (compare [15]) that having a portfolio of MIP solvers is beneficial.

4. Variants and extensions

Two main directions for modifications of the primal integral are (i) using a different base measure $p(t)$ and (ii) extending the integral function $P(T)$.

Concerning (i), this article has suggested taking the integral over a primal gap function defined by the current incumbent and a best known feasible solution. Analogously, we can define a *dual integral* by considering a dual gap function between the current and a best known dual bound. Using the "gap closed" (one of the first appearances was in 1985 [14]) as a measure for the performance of a cutting plane algorithm has the same pitfall as using the objective function of the best found solution for primal heuristics: it only considers the final state, ignoring the path that led there.

Taking the integral over the primal–dual gap as reported by most general-purpose MIP solvers can serve as a measure of their convergence speed. Again, this might be particularly worthwhile when a test set contains many instances which hit an imposed time limit. Reporting the gap at termination is prone to variations caused by bound changes around the time limit. Using a *primal–dual integral* instead reduces the impact of events that happen around the time limit: if as an extreme example one solver improves the bound after 3599 s and the other after 3601 s, the primal–dual integral will differ by less than 0.1%, whereas the gap after one hour can be arbitrarily different. Unlike the primal integral, a primal–dual integral does not even require the value of an optimal or best known solution as an input.

Concerning (ii), logarithmic scales are often used to put the focus on the factor between measured values rather than on their absolute difference. It can be argued for the time axis as well as for the gap axis that it makes sense to rather use a logarithmic scale, and incorporate the logarithm into the definition of $P(T)$.

We used primal heuristics for mixed integer linear programming as a showcase in this article, but never exploited specific structures of this problem class. The suggested measures can be used for any class of optimization problem which features a meaningful primal or dual bound, and any algorithm that produces a monotonic sequence of bound values. Similarly, instead of a gap function, any performance measure which evolves monotonically over time could be used.

5. Conclusion

The primal integral is less prone, though not immune, to common weaknesses of standard performance measures, notably the dependence on an (arbitrarily chosen) time limit. For two state-of-the-art MIP solvers, the primal integral changes by a factor of nearly 2 when disabling primal heuristics, whereas the running time to optimality only increases moderately in the same experiment.

We conclude that the primal integral and its variants (e.g., a primal–dual integral) are a valuable extension of the portfolio of available performance measures. We suggest taking it into account whenever analyzing algorithms that work particularly on the primal bound within an optimization procedure.

Acknowledgments

We thank Gregor Hendel, who provided the python scripts that compute the primal integral and plot the corresponding figures. We are further indebted to Hans Mittelmann, who made the log files available to us. Many thanks to Jonas Schweiger for proofreading a first draft of this paper, and to an anonymous reviewer for helpful suggestions. This research has been supported by the DFG Research Center Matheon *Mathematics for key technologies* in Berlin.

References

[1] T. Achterberg, SCIP: solving constraint integer programs, Mathematical Programming Computation 1 (1) (2009) 1–41.
[2] R. Bixby, M. Fenelon, Z. Gu, E. Rothberg, R. Wunderling, MIP: theory and practice—closing the gap, in: M. Powell, S. Scholtes (Eds.), Systems Modelling and Optimization: Methods, Theory, and Applications, Kluwer Academic Publisher, 2000, pp. 19–49.
[3] R.E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, R. Wunderling, Mixed integer programming: a progress report, in: The Sharpest Cut: the Impact of Manfred Padberg and His Work, in: MPS-SIAM Series on Optimization, vol. 4, 2004, pp. 309–326.
[4] CBC user guide—COIN-OR. http://www.coin-or.org/Cbc.
[5] H.P. Crowder, R.S. Dembo, J.M. Mulvey, Reporting computational experiments in mathematical programming, Mathematical Programming 15 (1978) 316–329.
[6] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Mathematical Programming 91 (2002) 201–213.
[7] FICO Xpress Optimizer. http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx.
[8] Gurobi optimization. http://www.gurobi.com/.
[9] A. Hoffman, M. Mannos, D. Sokolowsky, N. Wiegmann, Computational experience in solving linear programs, Journal of the Society for Industrial and Applied Mathematics 1 (1) (1953) 17–33.
[10] K.L. Hoffman, M. Padberg, Solving airline crew scheduling problems by branch-and-cut, Management Science 39 (6) (1993).
[11] J. Hooker, Needed: an empirical science of algorithms, Operations Research 42 (2) (1993) 210–212.
[12] IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.
[13] R.H.F. Jackson, P.T. Boggs, S.G. Nash, S. Powell, Guidelines for reporting results of computational experiments. Report of the ad hoc committee, Mathematical Programming 49 (1991) 413–425.
[14] E. Johnson, M. Kostreva, U. Suhl, Solving 0-1 integer programming problems arising from large scale planning models, Operations Research 33 (1985) 803–819.
[15] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R.E. Bixby, E. Danna, G. Gamrath, A.M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D.E. Steffy, K. Wolter, MIPLIB 2010, Mathematical Programming Computation 3 (2) (2011) 103–163.
[16] C.C. McGeoch, Toward an experimental method for algorithm simulation, INFORMS Journal on Computing 8 (1) (1996) 1–15.
[17] H. Mittelmann, Decision tree for optimization software: benchmarks for optimization software. http://plato.asu.edu/bench.html (accessed: 5.03.2013).
[18] G. Nannicini, P. Belotti, Rounding-based heuristics for nonconvex MINLPs, in: P. Bonami, L. Liberti, A. J. Miller, A. Sartenaer (Eds.), Proceedings of the EWMINLP, 2010, pp. 159–167.
[19] SoPlex. An open source LP solver implementing the revised simplex algorithm. http://soplex.zib.de/.