Mathematical Programming (2021) 188:523–537
https://doi.org/10.1007/s10107-020-01547-5

FULL LENGTH PAPER

Series B

The confined primal integral: a measure to benchmark heuristic MINLP solvers against global MINLP solvers

Timo Berthold[1] · **Zsolt Csizmadia**[2]

Received: 30 January 2020 / Accepted: 17 July 2020 / Published online: 29 July 2020

Abstract
It is a challenging task to fairly compare local solvers and heuristics against each other and against global solvers. How does one weigh a faster termination time against a better quality of the found solution? In this paper, we introduce the confined primal integral, a new performance measure that rewards a balance of speed and solution quality. It emphasizes the early part of the solution process by using an exponential decay. Thereby, it avoids that the order of solvers can be inverted by choosing an arbitrarily large time limit. We provide a closed analytic formula to compute the confined primal integral a posteriori and an incremental update formula to compute it during the run of an algorithm. For the latter, we show that we can drop one of the main assumptions of the primal integral, namely the knowledge of a fixed reference solution to compare against. Furthermore, we prove that the confined primal integral is a transitive measure when comparing local solves with different final solution values. Finally, we present a computational experiment where we compare a local MINLP solver that uses certain classes of cutting planes against a solver that does not. Both versions show very different tendencies w.r.t. average running time and solution quality, and we use the confined primal integral to argue which of the two is the preferred setting.

Keywords MINLP · Optimization software · Performance measure

Mathematics Subject Classification 90C30 Nonlinear programming

1 Introduction

When working with optimization software, there is a natural desire to compare different solvers to each other, with respect to both their speed and their solution quality.

✉ Timo Berthold
timoberthold@fico.com

1 Fair Isaac Germany GmbH, Takustr. 7, 14195 Berlin, Germany

2 Fair Isaac Europe Ltd, International Square, Starley Way, Birmingham B37 7GN, UK

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 [14]. 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 [10,15–17]. One key issue of the cited articles is the choice of suitable performance indicators. Common measures are the running time to find a first feasible solution, the running time to proven optimality, the objective value after a certain time limit, the number of branch-and-bound nodes (for branch-and-bound based algorithms) or the number of iterations (e.g., for interior point algorithms). Recently, Berthold suggested a measure called the *primal integral* [4] which combines the running time of an algorithm with the solution quality of a series of improving solutions it produces. The author used global solvers for mixed-integer linear programs as a showcase. Since the primal integral takes the development of the incumbent solution over time into account, it favors algorithms that find good solutions early. The author argues that it is less prone, though not immune, against the dependence on an (arbitrarily chosen) time limit, which is a weakness of many performance measures.

In the present paper, we suggest an extension of the primal integral which will diminish the time limit dependence even more. There are two major motivations for our work.

Our first motivation comes from taking the perspective of a user of optimization software. For a user, the early phases of a solution process are often perceived as the most important ones. The change in solution quality (and in the bound) during the first minutes appear more relevant than a change after several hours, even though both might have a comparable impact. This leads to the idea of scaling measures like the primal integral to put an emphasis on the early solution phases. In our experience, users of optimization software tend to think in orders of magnitudes: What can I achieve in a few seconds, what can I achieve in a few minutes, what can I achieve in a few hours? Ultimately, the solver developers do not know which time limits the solver users will employ, thus it is preferable to have a measure that is mostly independent of the time limit.

Our second, even stronger, motivation comes from the desire to compare heuristics and local solvers against global solvers. We will use mixed-integer nonlinear programming (MINLP) as a showcase. An MINLP is an optimization problem of the form

$$\min\{f(x) \mid \ g_k(x) \leq 0 \ \forall k \in \mathcal{K},\ l \leq x \leq u,\ x_j \in \mathbb{Z} \ \forall j \in \mathcal{I}\} \tag{1}$$

with objective function $f\colon \mathbb{R}^n \mapsto \mathbb{R}$, constraint functions $g_k\colon \mathbb{R}^n \mapsto \mathbb{R}$, $k \in \mathcal{K} := \{1, \ldots, m\}$, continuously differentiable, and possibly nonconvex, and variable bounds $l, u \in \overline{\mathbb{R}}^n$, where $\overline{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$. Furthermore, let $\mathcal{N} = \{1, \ldots, n\}$ be the index set of all variables and $\mathcal{I} \subseteq \mathcal{N}$ the set of variables that need to be integral in every feasible solution. We call an MINLP *convex* when all of its constraint functions g_k

are convex.[1] Otherwise, we call the MINLP *nonconvex*. Finally, an NLP is an MINLP with $\mathcal{I} = \varnothing$.

Even more than for other problem classes, local solvers are commonly used for MINLP, since global solvers are often not tractable for problems of practically relevant size and complexity. Loosely speaking, we call a solver global, if given infinite time and other resources, it will find a global optimum to a given optimization problem and prove its optimality. The differentiation between heuristic solvers and local solvers is more subtle, and not well established. Sometimes these terms are even used interchangeably.

For this article, we call a solver *local* if it may terminate with a suboptimal solution without hitting any user imposed limit (such as a time limit), but is designed to provide a form of local optimality guarantee at least in the case when certain regularity conditions are satisfied. Examples of this include that the KKT conditions are satisfied, optimality in the local continuous relaxation of the problem, or optimality once a certain set of variables is fixed to the computed solution values. As an example of a local solver consider Xpress-Nonlinear which performs a tree search over the discrete decision variables using sequential linear programming at each node of the search tree. It might terminate once the reference solution appears locally optimal or the error in the KKT conditions can no longer be reduced. At the same time, this locally optimal solution will typically not be globally optimal.

Finally, we call a solver *heuristic* when it may terminate with a suboptimal solution without hitting any user imposed limit and does not provide any additional dual information or follow any local optimality criterion. As an example, consider the various Feasibility Pump heuristics that have been proposed in the literature, for an overview see [7]. All of them perform a sequence of rounding and projection steps, producing two series of points and terminate, once these series coincide in a member. For those rounding and projection steps, constraints or integrality conditions are relaxed, new constraints are added to the problem and auxiliary objectives, like L1- and L2-norms are introduced. By throwing away the original objective of the problem, feasibility pump algorithms do not provide a local optimality condition w.r.t. this original objective.

When using the sheer running time as a measure, global solvers are at a clear disadvantage to local and heuristic solvers. The latter can always terminate prematurely and are not designed to explore the full search space, so they can by design be expected to run faster than a global solver. By choosing a large enough time limit, one can most likely make the global solver lose, once some problems are in the mix which are hard to solve to global optimality. Note that nevertheless, measuring running times is often seen in scientific publications when problem-specific heuristics are compared against MIP solvers like Xpress, Cplex, or Gurobi.

For a measure like the primal integral—designed to work well with heuristics—the question occurs of how to treat the nonzero gap at the point of termination w.r.t. the time limit. If we accounted the period between termination and time limit as constantly keeping the gap at termination, this would put the heuristic at a disadvantage. By choosing a large enough time limit, one could always make the global solver win

[1] This is a slight, but common, abuse of notation. Of course, the feasible set of a convex MINLP will be nonconvex in general, due to the discrete variables.

the comparison. If the gap was considered zero, the issue would be the same as with measuring running times. All of the mentioned points also hold for comparisons of local solvers or heuristics against other local solvers or heuristics, given that each of the solvers might use a very different trade-off of running time against solution quality.

In this paper, we suggest a new measure, the *confined primal integral* which incorporates an exponential decay in accounting the primal gap over time. This addresses both challenges: The perception of optimization software users and the tailing-off behavior that needs to be taken into account when comparing heuristics to global solvers.

2 The confined primal integral

To measure the quality of a given feasible solution for an MINLP against its known optimal solution, we define the *primal gap*.

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

$$\gamma_{\tilde{x}_{\text{opt}}}(\tilde{x}) := \begin{cases} 0, & \text{if } \left|f(\tilde{x}_{\text{opt}})\right| = |f(\tilde{x})| = 0, \\ 1, & \text{if } f(\tilde{x}_{\text{opt}}) \cdot f(\tilde{x}) < 0, \\ \frac{|f(\tilde{x}_{\text{opt}}) - f(\tilde{x})|}{\max\{\left|f(\tilde{x}_{\text{opt}})\right|, |f(\tilde{x})|\}}, & \text{else.} \end{cases}$$

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

Note that for two feasible MINLP solutions $\tilde{x}_1$, $\tilde{x}_2$ with $f(\tilde{x}_1) < f(\tilde{x}_2)$ and $\text{sign}(f(\tilde{x}_2)) = \text{sign}(f(\tilde{x}_{\text{opt}}))$ it holds that $\gamma_{\tilde{x}_{\text{opt}}}(\tilde{x}_1) < \gamma_{\tilde{x}_{\text{opt}}}(\tilde{x}_2)$.

Now assume that we have available the objective function values of intermediate incumbent solutions and the points in time when those have been found – for a given MINLP solver, a certain problem instance and a fixed computational environment. For defining the confined primal integral, we will consider the primal gap as a function over time, apply a time-dependent exponential scaling and compute the integral of that function.

Definition 2 Let $t_{\max} \in \mathbb{R}_{\geq 0}$ be a limit on the solution time of a MINLP solver. Its *primal gap function* $p\colon [0, t_{\max}] \mapsto [0, 1]$ is defined as:

$$p_{\tilde{x}_{\text{opt}}}(t) := \begin{cases} 1, & \text{if no incumbent until point } t, \\ \gamma_{\tilde{x}_{\text{opt}}}(\tilde{x}(t)), & \text{with } \tilde{x}(t) \text{ being the} \\ & \text{incumbent solution at point } t, \\ & \text{otherwise.} \end{cases}$$

and its *confined primal gap function* is defined as

$$\tilde{p}_{\tilde{x}_{\text{opt}},\alpha}(t) := p_{\tilde{x}_{\text{opt}}}(t) \cdot \exp(t/\alpha)$$

where $\alpha < 0$ is a scaling parameter.

For a fixed time limit $\mathrm{t}_{\max}$, a practical choice of α, due to its interpretability, is to set it to $\alpha = \mathrm{t}_{\max}/\log \iota$ with $\iota \in (0, 1)$ (hence, $\alpha < 0$). We will refer to the value ι as *importance* since, loosely speaking, it expresses how "important" a solution improvement at the time limit is considered in comparison to an improvement in the beginning of the solution process. E.g., when $\iota = 0.01$, the primal gap at the time limit will be divided by 100, which means that having a primal bound improvement close to the time limit would only have about 1% of the impact (per time) on the (yet to be defined) confined primal integral compared to what it would have had right in the beginning.

The confined primal gap function $p_{\tilde{x}_{\mathrm{opt}},\alpha}(t)$ is a non-smooth, piecewise exponential function with breakpoints whenever a new incumbent is found. It is strictly monotonically decreasing until the point at which the optimal solution is found and constant zero from that point on. In contrast, the primal gap function as described in [4] is a (non-strictly) monotonically decreasing step function.

A run of a solver in our sense essentially consists of a finite series of time stamps $T = t_1, t_2, \ldots, t_k$ corresponding to a time where a new incumbent solution has been found and its associated primal bound series $C = c_1, c_2, \ldots, c_k$ which represents the objective function value of the new incumbents being found at the respective points in time. For practical reasons, we extend both series by the elements $t_0 := 0$ and $c_0 := \infty$ to represent the initial state of the solver and by elements t_{k+1} and c_{k+1}. The latter might have two different interpretations which, however, can be treated identically from a theoretical point of view. Depending on the situation, the $(k+1)$-th elements might either represent the final state of the solver, in which case $t_{k+1} := \mathrm{t}_{\max}$ and $c_{k+1} := c_k$, or they might represent the situation that the solver finds a new incumbent solution during run time, in which case t_{k+1} corresponds to the elapsed run time and c_{k+1} to the new incumbent value, respectively.

Definition 3 Let $\mathrm{t}_{\max} \in \mathbb{R}_{\geq 0}$ be a time limit and let C be a series of incumbent solutions, together with a series of time stamps T with $t_{k+1} = \mathrm{t}_{\max}$. For a fixed scaling parameter $\alpha < 0$, we define the *confined primal integral (CPI)* $P_\alpha(T, C)$ of a run as:

$$P_\alpha(T, C) := \int_{t=0}^{\mathrm{t}_{\max}} \tilde{p}_{\tilde{x}_{\mathrm{opt}},\alpha}(t)\, dt = \alpha \sum_{i=1}^{k+1} p_{\tilde{x}_{\mathrm{opt}}}(t_{i-1})(\exp(t_i/\alpha) - \exp(t_{i-1}/\alpha)).$$

Figure 1 shows the difference between the "regular" and the confined primal integral. While all gap functions start at the same point $(0, 100\%)$, the confined gap functions constantly decrease, creating a larger and larger relative difference to the primal gap. At points t_1, t_2, and t_3, where new solutions are found, all functions have a discontinuity and "jump" to a lower level. From t_3 onwards, where the optimal solution has been found, all functions coincide in being constant zero. Different values of the importance value ι create different confined primal gap functions and hence different CPIs, as depicted by the green and blue areas. Each part of those areas can be computed analytically as the integral of an exponential function.

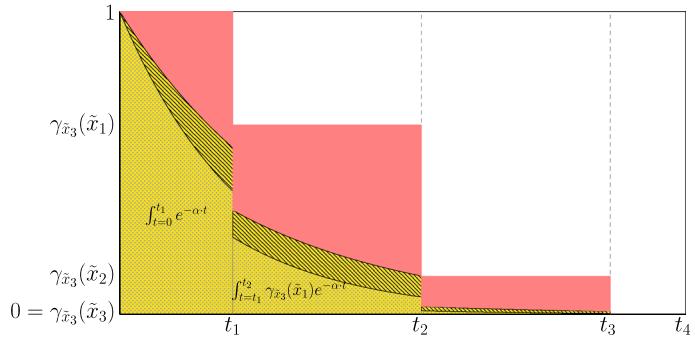

Fig. 1 Regular primal integral (red) and confined primal integral with two different importance values (yellow) (color figure online)

Note that we will always compute the CPI by extending the gap of the last found solution until the time limit, even if the underlying solution process stopped prematurely, e.g., because it was a local solver or a heuristic. The following lemma helps to put such a "penalty term" in perspective.

Lemma 1 *For any series* T, C*, any fixed* $\alpha < 0$ *and any time limit* t_{max}*,* $P_\alpha(T, C) \leq -\alpha$.

Proof It holds that $P_\alpha(T, C)$ is monotonically increasing in the time limit $\mathrm{t_{max}}$. Hence, it is sufficient to prove that $\lim_{\mathrm{t_{max}} \to \infty}(P_\alpha(T, C)) \leq -\alpha$ for any series T, C. Since $\gamma \in [0, 1]$, the estimation can be conducted independently of T, C, overestimating γ with a constant primal gap of 1. Therefore, $\lim_{\mathrm{t_{max}} \to \infty}(P_\alpha(T, C)) \leq \int_{t=0}^{\infty} exp(t/\alpha)\, dt = -\alpha$. □

The CPI being bounded independent of the time limit is a major difference to the original primal integral. An important consequence is that one cannot necessarily make a global solver (that will reach gap zero eventually) win against a local solver (that stopped with a nonzero gap) just by making the time limit arbitrarily large. This makes the extension of the final gap until the time limit reasonable since the additional penalty term can be expected to be comparably small, at least when the local solver found a reasonably good solution. Note that the CPI is a parametrized measure, with α being the degree of freedom.

As a numerical example consider the situation that we compare a heuristic and a global solver with the following solution processes. The heuristic starts, finds a 10% gap solution after one second, a 1% gap solution after ten seconds and terminates after a minute. The global solver starts, also finds a 10% gap solution after one second, but takes two minutes (instead of 10 seconds) to find a 1% gap solution and it finds a 0.8% gap solution after half an hour. Then it keeps running without improvement until an imposed time limit of two hours. In many cases, one would prefer the heuristic which is not only much faster in reaching a good solution, but also gives a definite termination before the global solver even found that solution. However, the classical primal integral would favor the global solver with a value of 72.9 compared to 73.99 for the heuristic.

With a very moderate choice of $\iota = 0.5$, the confined primal integral would favor the heuristic with a value[2] of 53.73 compared to 56.49 for the global solver. With a slightly more aggressive choice of $\iota = 0.1$, the difference would be 29.93 compared to 36.74, clearly favoring the heuristic. In fact, if we fixed the corresponding scaling factor of $\alpha = -3126$ and extended the time limit to an arbitrarily large value, the global solver could never win, since the confined primal integral of the heuristic is bounded by 33.06. In marked contrast, the classical primal integral could grow arbitrarily large.

Finally, this is the reason for choosing the term *confined* primal integral: Once the scaling factor α is determined, the value of the CPI is bounded by that very scaling factor, independent of the solution process and the time limit.

3 Calculating the confined primal integral

The CPI as defined above assumes, just like the primal integral in the original publication [4], the existence of a fixed reference optimal (or best known) solution. This is a reasonable assumption when thinking of fixed benchmark sets like MIPLIB [12] or MINLPLib [8], where such solution values are known in advance. However, when one wants to measure performance on new instances, in the closed formula given above, it is only possible to evaluate the CPI a posteriori, once all runs are finished. This might be considered impractical for a testing system where results are presented and stored immediately when they become available, as it is typically the case for commercial solver development.

Therefore, we further refine the notation of the confined primal integral and show how it can be dynamically updated. We define the *observed* CPI $P_\alpha^{\text{obs}}(T, C)$ analogously to $P_\alpha(T, C)$, with the restriction that we use the final incumbent c_k as reference, "best known" solution value $f(\tilde{x}_{\text{opt}})$. Further, we define the *correlated* CPI $P_\alpha^{\text{cor}}(T, C, \tilde{c})$ of run (T, C) and a solution value $\tilde{c} \in \mathbb{R}$ analogously to $P_\alpha(T, C)$, with the amendment that we use $\min(\tilde{c}, c_k)$ as reference solution value. This subtle distinction comes from the mindset that we might run a solver on a new, unknown problem instance for which we do not have a known optimal reference solution, hence we can only measure the observed CPI, and the reference solution will change during the run of the solver. When we run ℓ different solvers on such an instance, we would like to correlate the observations and update all observed CPIs to the same reference solution $\tilde{c} = \min(c_{k_1}, \ldots, c_{k_\ell})$, with $c_{k_1}, \ldots, c_{k_\ell}$ being the best found solution of each of the ℓ runs, hence we need the notation of a correlated CPI.

Two questions now arise: How can we update an existing observed CPI value "on the fly" when a new incumbent solution c_{k+1} is found? How can we consolidate several individual observed CPI values to correlated CPI values that all use the same reference solution value? Actually, the former case corresponds to a special case of the latter when setting $\ell = 1$, $\tilde{c} = \{c_{k+1}\}$ and interpreting the time point t_{k+1} as if it was the time limit.

It turns out, that such an update can be done independently of the exact solution sequence that led to the observed CPI, by using at most five attributes of the solution

[2] The following numbers are transcendental and have been rounded to four digits.

process, most of which are typically recorded anyway by commercial solvers. Those attributes are the (old) observed CPI, the (old) incumbent solution value, the new solution value, the time at which the new solution was found and in some cases a dynamic scaling factor. The update formula depends on the sign of the objective (and whether we are maximizing or minimizing, if we do not assume the optimization sense). For brevity of notation, we will use a series $\tilde{T} = \tilde{t}_0, \ldots, \tilde{t}_{k+1}$ with $\tilde{t}_i := exp(t_i/\alpha)$ for the terms that appear in the sum from Definition 3.

Lemma 2 *Consider a series $C = \{c_1, \ldots, c_k\}$ of incumbent solutions and a new incumbent c_{k+1} with $c_{k+1} < c_k$ being found at time t_{k+1}. If $c_1 < 0$ (and hence all other $c_i < 0$), then $P_\alpha^{cor}(t_{k+1}, C, \{c_{k+1}\}) = \frac{c_k}{c_{k+1}} P_\alpha^{obs}(t_k, C) + \alpha \cdot \tilde{t}_{k+1} \frac{c_{k+1} - c_k}{c_{k+1}}$. If $c_{k+1} > 0$ (and hence all other $c_i > 0$), then $P_\alpha^{cor}(t_{k+1}, C, \{c_{k+1}\}) = P_\alpha^{obs}(t_k, C) + D(c_k - c_{k+1})$ with a dynamic scaling factor $D = \alpha \sum_{i=2}^{k+1} \frac{\tilde{t}_i - \tilde{t}_{i-1}}{c_i}$.*

Proof See "Appendix". □

Note that the scaling factor D can be dynamically updated. Whenever a new incumbent is found, a new term needs to be added. Hence an MINLP solver only needs to store this single value D during the course of optimization, not a whole sequence of incumbent solution values and the points in time when they have been found. Furthermore, an update w.r.t. a new incumbent can be done in constant time.

We described how to update the CPI when adding new reference solutions. An important application for this is to compare runs without a fixed a-priori reference solution, but instead just using the better solution found in either of the runs. This then raises the question of how we can relate comparisons made with different reference solution to each other. Here, we observe that the CPI is transitive. This means that when we have three runs A, B, C and A has a smaller CPI than B (using the best solution of these two runs as a reference) and B has a smaller CPI than C (best solution of B, C as a reference), then A has a smaller CPI than C (best solution of A, C as a reference). This is a non-trivial observation, given that all the comparisons are made w.r.t. a different solution set. We formally state and prove it in the following proposition.

The important step in our proof of transitivity is to consider a different way of computing the CPI of a run.

Proposition 1 *Let three runs (T_A, C_A), (T_B, C_B) and (T_C, C_C) with best solution values c_a, c_b, c_c be given. If $P_\alpha^{cor}(T_A, C_A, c_b) < P_\alpha^{cor}(T_B, C_B, c_a)$ and $P_\alpha^{cor}(T_B, C_B, c_c) < P_\alpha^{cor}(T_C, C_C, c_b)$, then $P_\alpha^{cor}(T_A, C_A, c_c) < P_\alpha^{cor}(T_C, C_C, c_a)$.*

Proof Note that $c_0 = \infty$ requires a special handling when rolling out the definition of the confined primal gap function. According to Definition 2, the primal gap is considered to be 1 in this case.

Case 1, all objective values $c_i < 0$ for $i > 0$. First, we reformulate $P_\alpha^{\text{obs}}(T_A, C_A)$.

$$\begin{aligned}
P_\alpha^{\text{obs}}(T_A, C_A) &= \alpha\Big((\tilde{t}_1 - \tilde{t}_0) + \sum_{i=2}^{k+1}(\tilde{t}_i - \tilde{t}_{i-1})\frac{c_a - c_{i-1}}{c_a}\Big) \\
&= \alpha\Big((\tilde{t}_1 - \tilde{t}_0) + \sum_{i=2}^{k+1}(\tilde{t}_i - \tilde{t}_{i-1})\left(\frac{c_a}{c_a} - \frac{c_{i-1}}{c_a}\right)\Big) \\
&= \alpha\sum_{i=1}^{k+1}(\tilde{t}_i - \tilde{t}_{i-1}) - \alpha\sum_{i=2}^{k+1}(\tilde{t}_i - \tilde{t}_{i-1})\frac{c_{i-1}}{c_a} \\
&= \alpha(\tilde{t}_{k+1} - \tilde{t}_0) - \frac{\alpha}{c_a}\sum_{i=2}^{k+1}(\tilde{t}_i - \tilde{t}_{i-1})c_{i-1} \\
&= \int_0^{t_{\max}} \exp(t/s)\, dt - \frac{\alpha}{c_a} D_-^A
\end{aligned}$$

where $D_-^A = \sum_{i=1}^{k+1}(\tilde{t}_i - \tilde{t}_{i-1})c_{i-1} \leq 0$.

The value D_-^A is independent of the reference solution c_a; the same holds for D_-^B and D_-^C. Further, it holds that

$$\begin{aligned}
&P_\alpha^{\text{cor}}(T_A, C_A, c_b) < P_\alpha^{\text{cor}}(T_B, C_B, c_a) \\
&\Leftrightarrow \int_0^{t_{\max}} \exp(t/s)\, dt - \frac{\alpha}{\min(c_a, c_b)} D_-^A < \int_0^{t_{\max}} \exp(t/s)\, dt - \frac{\alpha}{\min(c_a, c_b)} D_-^B \\
&\Leftrightarrow D_-^A < D_-^B
\end{aligned}$$

The same argument implies that

$$P_\alpha^{\text{cor}}(T_B, C_B, c_c) < P_\alpha^{\text{cor}}(T_C, C_C, c_b) \Leftrightarrow D_-^B < D_-^C$$

From $D_-^A < D_-^B$ and $D_-^B < D_-^C$, it follows that $D_-^A < D_-^C$, which in turn proves Case 1.

Case 2, all objective values $c_i \geq 0$:

$$\begin{aligned}
P_\alpha^{\text{obs}}(T_A, C_A) &= \alpha\Big((\tilde{t}_1 - \tilde{t}_0) + \sum_{i=2}^{k+1}(\tilde{t}_i - \tilde{t}_{i-1})\frac{c_{i-1} - c_a}{c_{i-1}}\Big) \\
&= \alpha\Big((\tilde{t}_1 - \tilde{t}_0) + \sum_{i=2}^{k+1}(\tilde{t}_i - \tilde{t}_{i-1})\left(\frac{c_{i-1}}{c_{i-1}} - \frac{c_a}{c_{i-1}}\right)\Big) \\
&= \alpha\sum_{i=1}^{k+1}(\tilde{t}_i - \tilde{t}_{i-1}) - \alpha\sum_{i=2}^{k+1}(\tilde{t}_i - \tilde{t}_{i-1})\frac{c_a}{c_{i-1}} \\
&= \int_0^{t_{\max}} \exp(t/s)\, dt - c_a\alpha D_+^A
\end{aligned}$$

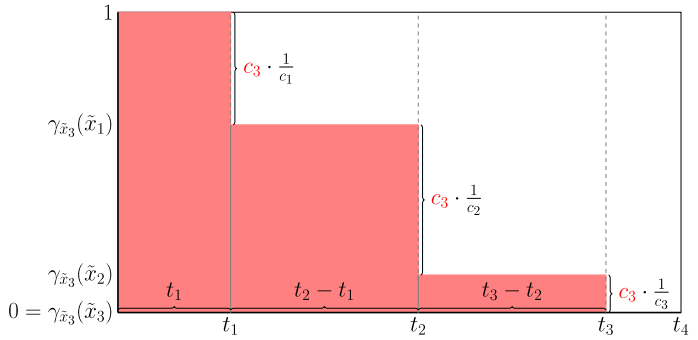

Fig. 2 For an update of the primal integral, it is easier to compute the white area (color figure online)

where $D_+^A = \sum_{i=1}^{k+1} (\tilde{t}_i - \tilde{t}_{i-1})/c_{i-1} \geq 0$. Again, the values D_+^A, D_+^B and D_+^C are independent of the reference solution and the same argument as in Case 1 can be made. As before, the case that the objective switches sign can be broken down into individual cases before and after the sign switch. □

Hence, we can compare independent test runs to each other while only storing three values per test run and instance: the final incumbent solution value, the observed CPI value and the scaling factor D. This is an important observation when there are many test runs (and instances) for which information is archived. In commercial optimization software development, these can easily be tens of thousands of test runs with thousands of instances per run. Here, having to store a dynamic list of incumbent solutions would be considered impractical.

Note that the proof of Proposition 1 gives rise to an alternative formula to update the CPI w.r.t. a new reference solution. Figure 2 sketches the idea for the "regular" primal integral without an exponential decay. The primal integral can be computed as the difference of the whole box ($\int_{t=0}^{\mathrm{t_{max}}} 100\% = \mathrm{t_{max}}$) and the white bars which are the difference between the current gap and 100%. As shown in the proof of Proposition 1, the rescaling of the white area can be done by multiplying with a factor that only depends on the old and new reference solution.

Also, it is a direct consequence from the proof of Proposition 1 that the CPI will increase when correlated to a strictly better reference solution:

Corollary 1 *Consider a series $C = \{c_1, \ldots, c_k\}$ of incumbent solutions and a new incumbent $c_{k+1} < c_k$. Then, $P_\alpha^{cor}(T, C, \{c_{k+1}\}) > P_\alpha^{obs}(T, C)$.*

Furthermore, a new reference solution will not change the order of two confined primal integrals.

Corollary 2 *Let three runs (T_A, C_A), (T_B, C_B) with best solution values c_a, c_b be given and $c_c < \min(c_a, c_b)$. If $P_\alpha^{cor}(T_A, C_A, c_b) < P_\alpha^{cor}(T_B, C_B, c_a)$, then $P_\alpha^{cor}(T_A, C_A, c_c) < P_\alpha^{cor}(T_B, C_B, c_c)$.*

There are various possibilities to extend the CPI, one of them is using a primal-dual gap. For MIP solvers and global MINLP solvers, it is common to report a primal-dual gap between the current incumbent solution and a dual bound that is, e.g., computed by

solving a continuous relaxation of the problem. The integral over the primal-dual gap can serve as a measure of their convergence speed and comes with the added advantage that it does not even require a priori knowledge of an optimal or best-known solution as an input. Using a confined primal-dual integral offers again the advantage that more emphasis is put on the early stages of the solution process. This makes it a valuable measure for MIP solving for applications where a quick convergence is key. It has been pointed out before that a portfolio of performance measures is good practice for optimization software development [5] and a confined primal-dual integral adds another point of view that nicely complements existing measures.

4 Computational experiments

We chose a comparison between two drastically different variants of a local solver as a test case to see whether the newly defined measure gives insights that are otherwise hard to get. The local solver that we chose is a pre-release version of FICO Xpress 8.9 [6]. For MINLP problems, Xpress implements a branch-and-bound algorithm based on a sequential linear programming [19] approach to solve NLP relaxations at every node, see also [3]. If there are nonconvex constraints in the MINLP, this will only provide a local optimum, making the approach a local solver. Branching will be conducted on integer variables with fractional values. No spatial branching is performed; leaf nodes without a fractional branching candidate will be pruned. Cuts are derived from the linearization at each node. On the one hand, those are the classical MINLP underestimator and approximation cuts [11]. On the other hand, one can generate standard MIP-cuts, like Gomory [13], flow-cover [18], knapsack cover [2], $\{0, 1/2\}$-cuts [9] etc. The separation of MIP-cuts has been added as a new feature for the solution of nonconvex MINLPs in Xpress 8.9 and we want to analyze in this computational study whether it is a worthwhile feature to activate by default. Hence, we compare versions of the nonlinear solver of Xpress with and without the separation of MIP-cuts at local nodes against each other. Linear underestimators and approximations for nonlinear functions will be separated in either version.

Like in the MIP case, we expect a significant speed-up of the solution process from using MIP cutting planes. However, they will typically not be globally valid; adding them to the problem might cut away the globally optimal solution or make the remaining problem infeasible. While this is acceptable behavior for a local solver, the question stands how much of an (assumed) speed-up is to be attributed to such rather undesired side effects and how the (assumed) loss in the solution quality can be weighed against the change in running time. Hence, we hypothesize that the confined primal integral is a suitable measure to use for comparing these two variants of a local solver and helps to resolve potentially contradicting results for running time and solution quality.

Our experiments were run on a cluster of identical machines equipped with Intel Xeon E5-2640 CPUs with 2.4 GHz and 64 GB of RAM. A time limit of 1800 seconds was set and each solve could use up to 20 threads. We used a test set of 795 MINLPs, which is a mix of publicly available instances, like those from MINLPLib [8], and customer instances. We used an importance value of $\iota = 10\%$. From the results we

Table 1 Aggregated computational results

	Running time (s)	Better obj	CPI
No MIP cuts	54.81	209:113	10.94
With MIP cuts	11.67	113:209	16.06
Factor	0.21	–	1.47

removed all instances where the solution path did not differ between the two runs. E.g., both paths will be identical when an integer solution is found at the root node before MIP cutting.

An aggregation of the results can be found in Table 1. Row no MIP cuts corresponds to a run with the Xpress control XSLP_CUTSTRATEGY set to 0, row with MIP cuts corresponds to a run with XSLP_CUTSTRATEGY set to 1. Columns running time and CPI show the shifted geometric mean [1] of the respective measures over all instances, using a shift value of 10. Column better obj counts the number of times that one run terminated with a solution that was at least 10% better than the best solution provided by the other run (including cases where only one version found a solution). Finally, Row factor shows the relative difference between the two settings. For running time and CPI, a factor larger than 1 means that no MIP cuts is better, a factor smaller than 1 means that with MIP cuts is better.

As expected, we see that adding MIP cuts during the MINLP tree search clearly improves the running time, by a factor of almost 5. At the same time, the solution quality clearly suffers. In roughly two out of three cases ($^{209}/_{323}$), the version without MIP cuts finds a better solution. Both performance measures show a significant difference between the two runs, but in opposite directions.

The question is standing whether the impressive gain in speed justifies finding a worse solution on many models. The CPI measure indicates that this might not be the case. It is 47% larger for the version that used MIP cuts during the MINLP search. While also giving a clear tendency, the CPI is less extreme than the other two measures, which is not surprising given that it takes aspects of both into account simultaneously. Interestingly, the picture did not change much when we increased the importance to $\iota = 50\%$ or decreased it to $\iota = 5\%$.

At this point the question arises, how a certain difference in the confined primal integral should be judged: Do we typically expect smaller or larger changes than for other performance measures? Due to the exponential scaling, the characteristic of the original primal integral representing the average solution quality is lost. What is true in either case is that the (confined) primal integral can be understood as scaled/weighted time, given that it is an integral of a function between zero and one over time. Hence, a relative change of the confined primal integral should be considered similarly significant as relative changes in solving time.

The confined primal integral was used to decide that XSLP_CUTSTRATEGY will be set to 0 by default in the Xpress 8.9 release. Furthermore, it is nowadays used as a standard measure to benchmark our MINLP solver in the daily working routine of the Xpress developer team.

5 Conclusion

We introduced the confined primal integral, a new performance measure which is an extension of the primal integral by Berthold [4]. We argued why it is particularly well-suited for comparing local and heuristic solvers against each other and against global solvers. Further, we proved that it is a transitive measure and presented an incremental updating formula that is independent of an a-priori knowledge of a reference solution. These two new results can be easily adapted for the original primal integral. In a brief computational experiment, using local solvers for nonconvex MINLP as a showcase, we demonstrated how the confined primal integral can help to compare two local solvers that show a very different balancing of solution quality versus running time.

We believe that the confined primal integral is a useful tool for computational experiments in nonlinear optimization and have made it part of the default performance evaluation criteria for the Xpress MINLP solver.

Acknowledgements We would like to thank the two anonymous reviewers for their constructive feedback that improved the quality of the paper a lot.

Appendix

Lemma 2 *Consider a series $C = \{c_1, \ldots, c_k\}$ of incumbent solutions and a new incumbent c_{k+1} with $c_{k+1} < c_k$ being found at time t_{k+1}. If $c_1 < 0$ (and hence all other $c_i < 0$), then $P_\alpha^{cor}(t_{k+1}, C, \{c_{k+1}\}) = \frac{c_k}{c_{k+1}} P_\alpha^{obs}(t_k, C) + \alpha \cdot \tilde{t}_{k+1} \frac{c_{k+1}-c_k}{c_{k+1}}$. If $c_{k+1} > 0$ (and hence all other $c_i > 0$), then $P_\alpha^{cor}(t_{k+1}, C, \{c_{k+1}\}) = P_\alpha^{obs}(t_k, C) + D(c_k - c_{k+1})$ with a dynamic scaling factor $D = \alpha \sum_{i=2}^{k+1} \frac{\tilde{t}_i - \tilde{t}_{i-1}}{c_i}$.*

Proof Case 1, $c_1 < 0$:

$$
\begin{aligned}
&P_\alpha^{\mathrm{cor}}(t_{k+1}, C, \{c_{k+1}\}) \\
&= \alpha \sum_{i=1}^{k+1} (\exp(t_i/\alpha) - \exp(t_{i-1}/\alpha)) p_{\tilde{x}_{\mathrm{opt}}}(t_{i-1}) \\
&= \alpha \Big(\tilde{t}_1 + \sum_{i=2}^{k+1} (\tilde{t}_i - \tilde{t}_{i-1}) \frac{c_{k+1} - c_{i-1}}{c_{k+1}} \Big) \\
&= \alpha \Big(\tilde{t}_1 + \sum_{i=2}^{k} (\tilde{t}_i - \tilde{t}_{i-1}) \frac{c_{k+1} - c_{i-1}}{c_{k+1}} + (\tilde{t}_{k+1} - \tilde{t}_k) \frac{c_{k+1} - c_k}{c_{k+1}} \Big) \\
&= \alpha \Big(\tilde{t}_1 + \sum_{i=2}^{k} (\tilde{t}_i - \tilde{t}_{i-1}) \frac{c_k - c_{i-1} + c_{k+1} - c_k}{c_{k+1}} + (\tilde{t}_{k+1} - \tilde{t}_k) \frac{c_{k+1} - c_k}{c_{k+1}} \Big) \\
&= \alpha \Big(\tilde{t}_1 + \sum_{i=2}^{k} (\tilde{t}_i - \tilde{t}_{i-1}) \frac{c_k - c_{i-1}}{c_{k+1}} + \sum_{i=1}^{k} (\tilde{t}_i - \tilde{t}_{i-1}) \frac{c_{k+1} - c_k}{c_{k+1}} + (\tilde{t}_{k+1} - \tilde{t}_k) \frac{c_{k+1} - c_k}{c_{k+1}} \Big)
\end{aligned}
$$

$$
\begin{aligned}
&= \alpha\Big(\tilde{t}_1 + \sum_{i=2}^{k}(\tilde{t}_i - \tilde{t}_{i-1})\frac{c_k - c_{i-1}}{c_{k+1}} + \tilde{t}_{k+1}\frac{c_{k+1} - c_k}{c_{k+1}}\Big) \\
&= \alpha\frac{c_k}{c_{k+1}}\Big(\tilde{t}_1 + \sum_{i=2}^{k}(\tilde{t}_i - \tilde{t}_{i-1})\frac{c_k - c_{i-1}}{c_k}\Big) + \alpha\cdot\tilde{t}_{k+1}\frac{c_{k+1} - c_k}{c_{k+1}} \\
&= \frac{c_k}{c_{k+1}}P_\alpha^{\text{obs}}(t_k, C) + \alpha\cdot\tilde{t}_{k+1}\frac{c_{k+1} - c_k}{c_{k+1}}
\end{aligned}
$$

Case 2, $c_{k+1} > 0$:

$$
\begin{aligned}
&P_\alpha^{\text{cor}}(t_{k+1}, C, \{c_{k+1}\}) \\
&= \alpha\Big(\tilde{t}_1 + \sum_{i=2}^{k+1}(\tilde{t}_i - \tilde{t}_{i-1})\frac{c_{i-1} - c_{k+1}}{c_{i-1}}\Big) \\
&= \alpha\Big(\tilde{t}_1 + \sum_{i=2}^{k}(\tilde{t}_i - \tilde{t}_{i-1})\frac{c_{i-1} - c_{k+1}}{c_{i-1}} + (\tilde{t}_{k+1} - \tilde{t}_k)\frac{c_k - c_{k+1}}{c_k}\Big) \\
&= \alpha\Big(\tilde{t}_1 + \sum_{i=2}^{k}(\tilde{t}_i - \tilde{t}_{i-1})\frac{c_{i-1} - c_k + c_k - c_{k+1}}{c_{i-1}} + (\tilde{t}_{k+1} - \tilde{t}_k)\frac{c_k - c_{k+1}}{c_k}\Big) \\
&= \alpha\Big(\tilde{t}_1 + \sum_{i=2}^{k}(\tilde{t}_i - \tilde{t}_{i-1})\frac{c_{i-1} - c_k}{c_{i-1}} \\
&\quad + (c_k - c_{k+1})\sum_{i=2}^{k}(\tilde{t}_i - \tilde{t}_{i-1})\frac{1}{c_{i-1}} + (\tilde{t}_{k+1} - \tilde{t}_k)\frac{c_k - c_{k+1}}{c_k}\Big) \\
&= P_\alpha^{\text{obs}}(t_k, C) + \alpha(c_k - c_{k+1})\sum_{i=2}^{k+1}(\tilde{t}_i - \tilde{t}_{i-1})\frac{1}{c_{i-1}}
\end{aligned}
$$

□

Note that in the above proof, all sums including a c_k start with 2, since the case $c_0 = \infty$ requires a special handling. According to Definition 2, the primal gap function is defined to be 1 in this case instead of being a difference of two objective function values divided by the larger one. Since $\tilde{t}_0 = 0$, the summand for $i = 0$ becomes $\tilde{t}_1$ in both cases of the proof, instead of $(\tilde{t}_i - \tilde{t}_{i-1})\frac{c_{k+1} - c_{i-1}}{c_{k+1}}$ in Case 1 or $(\tilde{t}_i - \tilde{t}_{i-1})\frac{c_{i-1} - c_{k+1}}{c_{i-1}}$ in Case 2.

Also note that the case that the incumbent solution switches the sign during optimization allows for an easy update, too. If the new incumbent at point t_{k+1} is the first one with a negative sign, then $P_\alpha^{\text{cor}}(T, C, \{c_{k+1}\}) = t_{k+1}$. If the sign switch happens at some point t_i with $2 < i < k + 1$, then the first part (until point i) of both primal integral sums is identical, and the second part (all summands greater i) can be updated as in Case 2 of the proof.

References

1. Achterberg, T.: SCIP: solving constraint integer programs. Math. Program. Comput. **1**(1), 1–41 (2009)
2. Balas, E.: Facets of the knapsack polytope. Math. Program. **8**(1), 146–164 (1975)
3. Belotti, P., Berthold, T., Neves, K.: Algorithms for discrete nonlinear optimization in FICO Xpress. In: 2016 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM), pp. 1–5. IEEE (2016)
4. Berthold, T.: Measuring the impact of primal heuristics. Oper. Res. Lett. **41**(6), 611–614 (2013)
5. Berthold, T.: Heuristic Algorithms in Global MINLP Solvers. Verlag Dr, Hut Munich (2014)
6. Berthold, T., Farmer, J., Heinz, S., Perregaard, M.: Parallelization of the FICO xpress-optimizer. Optim. Methods Softw. **33**(3), 518–529 (2018)
7. Berthold, T., Lodi, A., Salvagnin, D.: Ten years of feasibility pump, and counting. EURO J. Comput. Optim. **7**(1), 1–14 (2019)
8. Bussieck, M.R., Drud, A.S., Meeraus, A.: MINLPLib—a collection of test models for mixed-integer nonlinear programming. INFORMS J. Comput. **15**(1), 114–119 (2003)
9. Caprara, A., Fischetti, M.: {0, 1/2}-Chvátal–Gomory cuts. Math. Program. **74**(3), 221–235 (1996)
10. Crowder, H.P., Dembo, R.S., Mulvey, J.M.: Reporting computational experiments in mathematical programming. Math. Program. **15**, 316–329 (1978)
11. Duran, M.A., Grossmann, I.E.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical programming **36**(3), 307–339 (1986)
12. Gleixner, A., Hendel, G., Gamrath, G., Achterberg, T., Bastubbe, M., Berthold, T., Christophel, P., Jarck, K., Koch, T., Linderoth, J., et al.: MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. Tech. rep, Technical report, Optimization Online (2019)
13. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs and an algorithm for the mixed integer problem. In: 50 Years of Integer Programming 1958–2008, pp. 77–103. Springer (2010)
14. Hoffman, A., Mannos, M., Sokolowsky, D., Wiegmann, N.: Computational experience in solving linear programs. J. Soc. Ind. Appl. Math. **1**(1), 17–33 (1953)
15. Hooker, J.: Needed: an empirical science of algorithms. Oper. Res. **42**(2), 210–212 (1993)
16. Jackson, R.H.F., Boggs, P.T., Nash, S.G., Powell, S.: Guidelines for reporting results of computational experiments. Report of the ad hoc committee. Math. Program. **49**, 413–425 (1991)
17. McGeoch, C.C.: Toward an experimental method for algorithm simulation. INFORMS J. Comput. **8**(1), 1–15 (1996)
18. Padberg, M.W., Van Roy, T.J., Wolsey, L.A.: Valid linear inequalities for fixed charge problems. Oper. Res. **33**(4), 842–861 (1985)
19. Palacios-Gomez, F., Lasdon, L., Engquist, M.: Nonlinear optimization by successive linear programming. Manag. Sci. **28**(10), 1106–1120 (1982)