Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

GREGOR HENDEL[1]

Enhancing MIP branching decisions by using the sample variance of pseudo-costs

[1]funded by the German Federal Ministry of Education and Research (fund number 05M14ZAM) within the Research Campus Modal.

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Enhancing MIP branching decisions by using the sample variance of pseudo-costs

Gregor Hendel

November 26, 2014

Abstract

The selection of a good branching variable is crucial for small search trees in Mixed Integer Programming. Most modern solvers employ a strategy guided by history information, mainly the variable pseudo-costs, which are used to estimate the objective gain. At the beginning of the search, such information is usually collected via an expensive look-ahead strategy called strong-branching until variables are considered reliable.

The reliability notion is thereby mostly based on fixed-number thresholds, which may lead to ineffective branching decisions on problems with highly varying objective gains.

We suggest two new notions of reliability motivated by mathematical statistics that take into account the sample variance of the past observations on each variable individually. The first method prioritizes additional strong-branching look-aheads on variables whose pseudo-costs show a large variance by measuring the relative error of a pseudo-cost confidence interval. The second method performs a two-sample Student-t test for filtering branching candidates with a high probability to be better than the best history candidate.

Both methods were implemented in the MIP-solver SCIP and computational results on standard MIP test sets are presented.

1 Introduction

A *Mixed Integer Program (MIP)* denotes a minimization problem of a linear objective function under linear inequalities and integrality restrictions for a subset of the variables, or to prove that no solution exists. We use the term "mixed" to refer to the occurence of two variable types, continuous and integral variables, in the problem formulation.

Most modern solvers for MIP [CBC, CPL, XPR, GUR, SCI] apply a *branch-and-bound* procedure [Dak65, LD60], which creates a search tree for a MIP P by a successive problem division based on the LP-relaxation information at a node. In the most common scheme of variable-based branching, it is crucial to select good branching variables in order to quickly reach terminal nodes and thus keep the required search tree small. A *branching rule* is a scoring mechanism to guide the selection of a branching variable at each inner node of the search tree.

Branching rules [AB09, BGG$^+$71] using variable history information of prior branching decisions have been shown to perform well at later stages of the search, see also [LS99]. The initial lack of information can be overcome by

a computationally expensive *strong-branching*-initialization [ABCC95], which virtually performs a 1-level look-ahead by solving the child node LP-relaxations for a subset of the fractional variables and then selects the best candidate.

The current state-of-the-art branching rule for balancing between strong-branching and estimation, *reliability-branching* [AKM04], uses a fixed number of branching decisions after which the variable information is considered *reliable*. This approach has the disadvantage that it uses the same fixed reliability threshold for all variables. In practice, however, it appears natural that variables that are structurally different inside a MIP model also have different reliability requirements. Another disadvantage of a fixed parameter is that it might not scale well with increasing problem size.

The aim of the present paper is to introduce different notions of reliability by exploiting more statistical information during the process of (strong-)branching. Using the sample variance of past observations, we formulate two criteria for switching between strong-branching and estimation that take into account each variable history individually. We perform computational experiments on standard MIP test sets to evaluate the impact of our approach.

The remainder of this article is organized as follows: First, we summarize past and recent related work by other authors from the literature in Section 2. Section 3 introduces the necessary notation and presents the reliability branching rule in more detail. Afterwards, we introduce new notions of reliability in Section 4, and present computational results, which were obtained with an implementation in the Constraint Integer Programming framework SCIP [SCI] in Section 5. We finish with some concluding remarks in Section 6. The appendix contains an instance-wise summary of our computational experiments.

2 Related Work

Research on branching rules for Mixed Integer Programming has been a focus of interest since the advent of the Branch-and-Bound procedure in the 1960's [Dak65, LD60]. Note that in this paper, we only consider variable-based branching. This concept is generalizable by incorporating branching on general disjunctions, which was introduced in [RF81].

Pseudo-costs, which measure the average objective gain for every integer variable, and their use for branching first appeared in [BGG$^+$71]. The use of degradation bounds for the pseudo-cost initialization was suggested in [GR77]. Equipped with more computational power, strong-branching was first applied in the context of the Traveling Salesman Problem [ABCC95], whereas its first use for general MIP solving is attributed to the commercial MIP solver CPLEX [CPL]. An important computational study for these techniques, also in the context of node selection, can be found in [LS99].

Recently, Gamrath [Gam13] improved the strong-branching procedure by also applying domain propagation techniques at each sub-node during strong-branching. Furthermore, Berthold et al. [BGS14] proposed cloud branching to overcome the degeneracy of LP-relaxation solutions by considering variable fractionalities as intervals rather than points. The computational complexity for this approach is comparable to the effort of strong-branching because 2 sub-LP-relaxations have to be solved for every variable. In another recent work [FM12], Fischetti and Monaci observe unnecessary strong-branching effort at the pres-

ence of *chimerical* variables, i.e. fractional variables with little or no effect on the objective of the LP solutions. They exploit this fact to safely ignore such candidates for the strong-branching procedure.

The pseudo-cost branching rule is an effective replacement of the strong-branching rule at later stages of the search but lacks information at the beginning. For that reason, combinations of pseudo-cost branching and strong-branching have been developed that either use a single strong-branching initialization on uninitialized variables, and pseudo-costs for every initialized candidate, or strong-branching at the topmost d levels of the tree, and pseudo-cost branching at deeper levels. The state-of-the-art branching scheme, which is applied by most modern MIP solvers albeit the concrete implementation might vary, is *reliability branching* [AKM04], see also Section 3. A threshold number is dynamically adjusted at every node depending on the proportion of LP iterations during strong-branching and the total number of Simplex iterations spent during solving regular nodes, see also [Ach07] for further details. Other forms of history information such as inference or cutoff histories have been adopted for general MIP in [Ach09]. *Hybrid reliability branching* [AB09] combines pseudo-costs and deduction-based history information into a single score.

For recent variable branching methods that use other techniques than history information, see, e.g., [GS11, KKNS09]. In [PC11], Pryor and Chinneck presented a branching strategy for quickly finding feasible solutions that approximates solution densities by means of normal distributions. Although their approach is quite different from the one presented here, their work has indeed been a motivation to further study links between statistics and optimization. Fischetti and Monaci [FM11] recently presented a method for restricting the set of branching candidates by calculating so-called *backdoor sets* in advance.

The approach presented here uses variations of past branching information for the decision if strong-branching should be continued on a variable or not. It extends the idea of reliability branching by taking into account each variable individually. In the present paper, we further concentrate only on pseudo-costs and do not consider other history information. We do not collect any information prior to the actual search as in [KKNS09, FM11]. Finally, it should be noted that in general there is no containment relation between the variable subsets considered by reliability branching with fixed number thresholds and our approach, i.e. neither is a strict subset of the other.

3 Reliability Branching with fixed-number thresholds

We call an optimization problem of the form

$$c^{\text{opt}} := \inf\left\{c^t x \,:\; Ax \le b,\; l \le x \le u,\; x \in \mathbb{R}^n,\; x_j \in \mathbb{Z} \text{ for all } j \in \mathcal{I}\right\}, \quad \text{(MIP)}$$

Mixed Integer Program (MIP) and denote by c^{opt} the *optimal objective value* of a MIP. Furthermore, $c \in \mathbb{R}^n$ is called *cost vector*, and $A \in \mathbb{R}^{m,n}$ and $b \in \mathbb{R}^m$ represent the set of linear inequalities of a given MIP. By $l, u \in \mathbb{R}^n_\infty$, we denote *bound requirements* for the variables, and use a subset $\mathcal{I} \subseteq \{1, \ldots, n\}$ of the variables index set to formulate *integrality restrictions* and call variables indexed by $j \in \mathcal{I}$ *integral variables*. If the set of integrality restrictions is empty, we call

(MIP) a *Linear Program (LP)*. An LP $\tilde{P}$ is called the *LP-relaxation* of a MIP P if it is derived from P by dropping the integrality restrictions of P. Since the solution space of $\tilde{P}$ is a superset of the solution space of P, its holds that $c_{\tilde{P}}^{\text{opt}} \leq c_P^{\text{opt}}$.

The *branch-and-bound* procedure [Dak65, LD60] creates a search tree for a MIP $P =: P^{(0)}$ by a successive problem division called *branching* based on the LP-relaxation information at a node. Let $P^{(l)}$ be a feasible (sub-)problem currently processed. We solve the LP-relaxation of $P^{(l)}$ and obtain an LP-solution $\tilde{y}$ with objective value $c^t\tilde{y} = \tilde{c}_{P^{(l)}}$. If $\tilde{y}$ violates some of the integrality restrictions $\mathcal{F} \subseteq \mathcal{I}$, branching creates two child problems $P_-^{(l)}, P_+^{(l)}$ by selecting a *fractional variable* $j \in \mathcal{F}$ and locally restricting the lower and upper bound of j in the child problems to $u_j \leftarrow \lfloor \tilde{y}_j \rfloor$ in $P_-^{(l)}$ and $l_j \leftarrow \lceil \tilde{y}_j \rceil$ for $P_+^{(l)}$, respectively. Either restrictions renders $\tilde{y}$ infeasible. The created problems are then enqueued in a list of open subproblems. The procedure terminates when there is no open subproblem left.

For a fractional variable $j \in \mathcal{F}$ we define its *up-fractionality* and *down-fractionality* as

$$f_j^+ := \lceil \tilde{y}_j \rceil - \tilde{y}_j \quad \text{and} \quad f_j^- := \tilde{y}_j - \lfloor \tilde{y}_j \rfloor,$$

respectively. The decision on which fractional variable to branch is crucial for the success of the branch-and-bound search. A branching rule is characterized by its *score function* $\vartheta : \mathcal{F} \to \mathbb{R}$. It selects as branching variable some $j^* \in \mathcal{F}$ with $\vartheta(j^*) \geq \vartheta(j)$ for all $j \in \mathcal{F}$. In this paper, branching scores $\vartheta^-(j)$ and $\vartheta^+(j)$ are calculated separately for the two branching directions and combined afterwards by taking their *product score*

$$\vartheta(j) := \max\{\vartheta^+(j), \epsilon\} \cdot \max\{\vartheta^-(j), \epsilon\} \tag{1}$$

with a small $\epsilon = 10^{-6}$. The use of the product was proposed in [AKM04] in order to find a good balance between the sizes of the resulting subtrees.

Throughout this paper, we will give definitions and explanations only for the down-branch. The according formula and argumentation for the up-direction can be derived analogously.

Let $P_-(j)$ denote the MIP obtained by branching down on $j \in \mathcal{F}_P$. In this paper, we focus on the *gain* in the objective function

$$\vartheta^-(j) = \tilde{c}_{P_-(j)} - \tilde{c}_P \tag{2}$$

in the child node LP-relaxation objectives w.r.t. their parent as branching score.

Since this information is unknown by the time a candidate needs to be selected, the strong-branching rule determines $\vartheta_{\text{str}}^-(j)$ and $\vartheta_{\text{str}}^+(j)$ by virtually solving $2 \cdot |\mathcal{F}|$ child node relaxations and evaluating the gains (2). Although strong-branching is guaranteed to select the locally best candidate regarding the objective gain, the exhaustive solving of child nodes often makes the computational cost of this procedure prohibitive. However, it is well suited as an initialization method for pseudo-costs.

The *pseudo-costs* [BGG+71] of a variable are a typical measure to estimate its impact on the children objective gain. Consider a node P with LP solution value $\tilde{c}_P$ and a fractional variable $j \in \mathcal{F}_P$. Let $P_-^{(j)}$ be the down-child of P whose

LP-relaxation was solved to optimality. The normalization of the objective gain between $P_-^{(j)}$ and P,

$$\varsigma_j^-(P) := \frac{\tilde{c}_{P_-} - \tilde{c}_P}{f_j^-}$$

by the fractionality of j in $\tilde{y}^P$ is called *unit gain*. The pseudo-costs of a variable are the average over all such unit gains,

$$\Psi_j^- := \begin{cases} \frac{\gamma_j^-}{\eta_j^-}, \text{ if } \eta_j^- > 0, \\ 0, \text{ else,} \end{cases} \tag{3}$$

where η_j^- denotes the number of problems Q for which j was selected as branching variable and the child node $Q_-^{(j)}$ has been solved and was feasible, and γ_j^- the sum of obtained unit gains over all these problems. If η_j^- is 0, we call j *uninitialized* in this direction. The *pseudo-cost score function* uses the pseudo-cost information

$$\vartheta_{\text{ps}}^-(j) := \Psi_j^- \cdot f_j^-$$

to estimate the objective gain in the child obtained by branching on j.

We give a definition of reliability branching that is more general than the original definition by Achterberg et al. [AKM04]:

Definition 1 (General Reliability branching) *Let P be a MIP with non-empty set of fractionals $\mathcal{F}$. Given a subdivision $\mathcal{F} = \mathcal{F}^{rel} \dot\cup \mathcal{F}^{url}$ of the fractionals into* reliable *and* unreliable *candidates, we define the* general reliability branching score function *of $j \in \mathcal{F}$ as*

$$\vartheta_{rel}^-(j) := \begin{cases} \vartheta_{str}^-(j), & \text{if } j \in \mathcal{F}^{url}, \\ \vartheta_{ps}^-(j), & \text{if } j \in \mathcal{F}^{rel}. \end{cases} \tag{4}$$

General reliability branching performs strong-branching on the set of unreliable candidates $\mathcal{F}^{\text{url}}$ to determine their exact gains (2).

A reliability branching rule is characterized by its notion of (un-)reliability. We refer to the notion of reliability by Achterberg et al. [AKM04], as *fixed-number threshold reliability*:

Definition 2 (Fixed-number threshold reliability) *Given a reliability parameter $\eta > 0$,* fixed-number threshold (fnt)-reliability *splits the fractionals according to*

$$\mathcal{F}_{fnt}^{url}(\eta) := \{j \in \mathcal{F} : \ \min\{\eta_j^-, \eta_j^+\} < \eta\} \tag{5}$$

We call a variable $j \in \mathcal{F} \setminus \mathcal{F}_{fnt}^{url}(\eta)$ (fnt)-reliable.

Using the term "fixed-number", we emphasize that (fnt)-reliability of a variable solely depends on the number of previous branching observations. Achterberg et al. [AKM04] suggested to use 8 as threshold, currently, SCIP uses 5. In the next section, we introduce novel notions of reliability.

4 Relative-error- and hypothesis-reliability

The drawback of (fnt)-reliability is that a fixed threshold is supposed to measure the reliability of all variables of the problem equally well. Intuitively, it seems desirable to have a more individual look at the pseudo-cost information of every variable and to continue strong-branching on those candidates whose pseudo-costs fail to converge. In the following, we extend the statistical model for pseudo-costs by including the sample variance, which allows for the construction of confidence intervals and testing of hypotheses. There are many textbooks that cover these topics in more detail, see, e.g. [Rou14].

We model the unit gains of a variable $j \in \mathcal{I}$ as samples of a normally distributed random variable $C_{j,-} \sim \mathcal{N}(\mu_{j,-}, \sigma^2_{j,-})$ with unknown *mean* $\mu_{j,-}$ and *variance* $\sigma^2_{j,-}$. The pseudo-costs represent an estimate for $\mu_{j,-}$. By using the *corrected sample variance*, we obtain an estimate for the variance, as well:

Definition 3 *Let $X_1, \ldots, X_n$ be independent, identically distributed samples. The* corrected sample variance *about the sample mean $\bar{X}$ is given by*

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2 = \frac{1}{n-1} \sum_{i=1}^{n} X_i^2 - \frac{1}{n(n-1)} \left(\sum_{i=1}^{n} X_i\right)^2 \tag{6}$$

The corrected sample variance is an unbiased estimate of the variance of the underlying distribution of the X_i. The right term of Equation (6) allows for constant-time updates of s^2 every time a new sample X is observed.

With increasing n, we can expect $\bar{X}$ to approach the mean of the distribution from the law of large numbers. Under the assumption that the samples $X_1, \ldots, X_n$ are drawn from a normal distribution with unknown mean μ and variance σ^2, the random variable

$$T := \frac{\bar{X} - \mu}{s/\sqrt{n}}$$

is distributed along a Student's t-distribution with $n-1$ degrees of freedom. This relation can be used to construct a a *confidence interval* I, which contains the true value of μ with a probability of $1-\alpha$ for any *error rate* $0 < \alpha < 1$:

$$I = \left[\bar{X} - t_{\alpha,n-1} \frac{s}{\sqrt{n}}, \bar{X} + t_{\alpha,n-1} \frac{s}{\sqrt{n}}\right],$$

denoting by $t_{\alpha,n-1} > 0$ the α-percentile of the distribution of T. The distance of the endpoints of I relative to its center $\bar{X} \neq 0$,

$$\epsilon^{\text{rel}} = t_{\alpha,n-1} \cdot \frac{s}{\sqrt{n}|\bar{X}|}, \tag{7}$$

is called the *relative error* of the estimation.

4.1 Relative-error-reliability

Applied to pseudo-costs, we determine the relative error for the pseudo-costs associated with each variable. Whenever a new unit gain for variable $j \in \mathcal{F}$ in the down-branching direction at a node P was observed, we increase the counter

η_j^- by 1 and update the sum of unit gains γ_j^-. In addition, we keep track of the sum of squared unit gains $(\varsigma_j^-(P))^2$. This enables us to calculate the sample variance $\left(s_j^-\right)^2$ whenever $\eta_j^- \geq 2$. At a node Q, we calculate the relative error ϵ_j^- of the current pseudo-costs Ψ_j^- as

$$\epsilon_j^- := 1.96 \cdot \frac{s_j^-}{\sqrt{\eta_j^-}\Psi_j^-}. \tag{8}$$

In (8), we substitute $t_{\alpha,n-1}$ from (7) by the constant 1.96, which represents the limit α-percentile $\lim_{\eta_j^- \to \infty} t_{\alpha,\eta_j^- - 1}$ for $\alpha = 0.05$. Thus, we slightly underestimate the relative error of the pseudo-cost at a confidence level of 95 %. Recall that pseudo-costs are always non-negative. Hence, we can omit the absolute in the denominator of (7). Furthermore, if the upwards pseudo-costs of j are equal to zero, this also holds for the sample variance $\left(s_j^-\right)^2$. We therefore set the relative error to zero in this case.

Definition 4 (Relative-error-reliability) *For $\eta > 0$,* relative-error (rer)-reliability *splits the fractionals according to*

$$\mathcal{F}_{rer}^{url}(\eta) := \{j \in \mathcal{F} : \ \max\{\epsilon_j^+, \epsilon_j^-\} \geq \eta\}. \tag{9}$$

We call a variable $j \in \mathcal{F} \setminus \mathcal{F}_{rer}^{url}(\eta)$ (rer)-reliable.

The rationale of (rer)-reliability is to continue strong-branching on the subset of variables with highly varying objective gains, whereas variables with constant gains are early considered (rer)-reliable. In order to obtain relative errors for the branching directions, we need at least $\eta_j^-, \eta_j^+ \geq 2$ observations in each direction. Note that a variable, which has already been (rer)-reliable, can become (rer)-unreliable again when the relative error rerises above the threshold after new information becomes available. In Section 5, we test an implementation of (rer)-reliability branching.

4.2 Hypothesis-reliability

The disadvantage of (rer)-reliability is that it is likely to spend much strong-branching effort on variables with overall low objective gains, but high relative error. In order to overcome this, it is possible to restrict the variables that are selected for strong-branching evaluation to only candidates with a probability to be actually better than the best candidate j^{ps} according to pseudo-cost branching. Roughly speaking, we want to ensure that there is little probability that $f_j^- \mu_j^- > f_{j^{\text{ps}}}^- \mu_{j^{\text{ps}}}^-$.

Therefore, we test against the hypothesis that a fractional $j \in \mathcal{F}$ has an objective gain at least as high as j^{ps}, i.e., $f_j^- \mu_{j,-} \geq f_{j^{\text{ps}}}^- \mu_{j^{\text{ps}},-}$. For two variables $i, j \in \mathcal{F}$ with fractionalities f_i^- and f_j^-, we use the *pooled variance*

$$S_{i,j}^- := \frac{(\eta_i^- - 1)(f_i^-)^2 \left(s_i^-\right)^2 + (\eta_j^- - 1)(f_j^-)^2 \left(s_j^-\right)^2}{\eta_i^- + \eta_j^- - 2}$$

to calculate a *2-sample t-value* for i and j,

$$T^-_{i,j} := \sqrt{\frac{\eta^-_i \eta^-_j}{\eta^-_i + \eta^-_j}} \frac{f^-_i \Psi^-_i - f^-_j \Psi^-_j}{S^-_{i,j}}.$$

Under the hypothesis, $T^-_{j^{\text{ps}},j}$ follows a Student-t distribution with $\eta^-_{j^{\text{ps}}} + \eta^-_j - 2$ degrees of freedom. If, for a given threshold $0 < \alpha < 1$, $T^-_{j^{\text{ps}},j}$ exceeds $t^-_{\alpha,j^{\text{ps}},j} := t_{\alpha,\eta^-_{j^{\text{ps}}}+\eta^-_i-2}$, we can reject the hypothesis with an error probability of at most $\alpha/2$. The division by two is justified because the hypothesis is one-sided. Conversely, if the hypothesis cannot be safely rejected, it is safer to perform strong-branching on the two candidates.

The second novel notion of reliability in the present paper rules out fractional variables with little probability to be better than the best pseudo-score candidate:

Definition 5 (Hypothesis-reliability) *Let $j^{ps} \in \mathcal{F}$ be the best pseudo-cost fractional candidate for branching, and let $0 < \alpha < 1$ be a rejection probability. The unreliable fractional set for* hypothesis-reliability *is*

$$\mathcal{F}^{url}_{hyp}(\alpha) := \left\{ j \in \mathcal{F} \,:\, T^-_{j^{ps},j} < t^-_{\alpha,j^{ps},j} \text{ and } T^+_{j^{ps},j} < t^+_{\alpha,j^{ps},j} \right\}. \tag{10}$$

Variables $j \in \mathcal{F} \setminus \mathcal{F}^{url}_{hyp}(\alpha)$ are called (hyp)-reliable.

For practical reasons, we also include variables j with $\min\{\eta^-_j, \eta^+_j\} \leq 1$. It should be noted that the best pseudo-cost candidate j^{ps} is never (hyp)-reliable because $T^-_{j^{\text{ps}},j^{\text{ps}}} = T^+_{j^{\text{ps}},j^{\text{ps}}} = 0$. If no other candidate than j^{ps} is (hyp)-unreliable, this means that no other fractional variable has an estimated objective gain nearly as good as j^{ps}. In this case, we immediately branch on j^{ps} without strong-branching. In the experiments in the following section, we tested an error probability of $\alpha = 0.2$, i.e. the error probability for ruling out a better candidate based on the current branching history is $\alpha/2 = 10\,\%$.

5 Computational results with SCIP

We implemented the new reliability notions from Section 4 into the existing reliability branching rule of a development version of the Constraint Integer Programming framework SCIP [SCI] version 3.1.0.2, which we compiled with a gcc compiler version 4.8.2. As underlying LP-solver, we used SOPLEX [SOP] version 2.0. We used SCIP with default settings except for the following changes. For using a pure objective-based branching score function as in Section 3, tie-breakers such as, e.g., inference scores were deactivated by setting their corresponding weight to 0. Furthermore, we set the known optimal solution values – in case they exist – minus a small threshold 10^{-9} as objective cutoffs, so that only a proof for the optimality/infeasibility of a problem needed to be found. We also disabled all primal heuristics and activated depth-first search node selection as an attempt to minimize performance variability [Dan08, KAA+11] due to other factors than the tested branching rules. Finally, the child node selection was changed to use solely pseudo-costs, where SCIP with default settings uses a hybrid approach together with inference scores.

The test bed for our comparison of the different approaches consists of a subset of instances from the three publicly available libraries MIPLIB 3.0 [BCMS98], MIPLIB 2003 [AKM06], and MIPLIB 2010 [KAA+11], from which we omitted four instances for which an optimal objective value is not known by the time of this writing. Since we are mainly interested in reducing the search tree size, we further dropped all 29 instances that could be solved before or during the processing of the root node. Our final test bed thus contains 135 MIP instances.

The computations were performed on a cluster of 32 computers, each of which runs with a 64bit Intel Xeon X5672 CPUs at 3.20 GHz with 12 MB cache and 48 GB main memory. The operating system was Ubuntu 14.4. Hyperthreading and Turboboost were disabled. We ran only one job per computer in order to minimize the random noise in the measured running time that might be caused by cache-misses if multiple processes share common resources. Finally, all experiments were run with a time limit of 2h and a 40 GB memory limit.

The newly proposed notions of reliability from Section 4 are represented by four different settings: `(hyp)` renders candidates (hyp)-unreliable according to the rule (5), whereas `(rer)-0.01`, `(fnt)-5`, and `(rer)-0.1` use (rer)-reliability regarding relative errors in pseudo-cost confidence intervals at three different threshold levels 1 %, 5 %, and 10 %. We compare them to (fnt)-reliability at a fixed threshold of 5, denoted by `(fnt)-5`. The latter setting constitutes the default of SCIP except that we disabled the threshold to be dynamically adjusted during the search.

In this section, we only present compressed results of our experiments. For an instance-wise outcome, please refer to Tables 5 and 6 in the Appendix. The first three tables show the aggregated results regarding the solving time t (sec) and the number of explored search tree nodes n for all instances and for only those which could be solved within the time limit by all settings. We consider node results incomparable between settings where the solution status differs and thus only show time results for all instances. We report shifted geometric means with a shift of 10 seconds and 100 nodes, respectively. The column "%" shows the percentage deviation from the result for the reference setting `(fnt)-5`; values below 100 represent an improvement in this respect.

In Table 1, we compare the results over all instances from the test bed. 98 instances could be solved by all settings within the time limit of 2h, for which the reference run was fastest regarding the solving time, but also required the most branch-and-bound nodes on average. The highest node reduction of 19.5 % was obtained with the setting `(rer)-0.01`. For our novel notion of (rer)-reliability, the different thresholds influence the node reduction as one might have expected: increasing the tolerance level causes a larger number of nodes. By using (hyp)-reliability, we obtained a node reduction of 17.6 %, which is better than for all other settings except for `(rer)-0.01`. The latter setting also shows an increase in the running time of 6 % compared to the reference run, whereas the setting `(hyp)` was almost performance neutral regarding the running time.

Table 2 contains only instances for which at least one of the settings needed more than 1000 nodes before termination. With (hyp)-reliability, we could improve the performance of SCIP w.r.t. the reference run by 28.6 % nodes and also obtain a slight time reduction in total, whereas the time on instances in the left group increased by 1.7 %. With (hyp)-reliability, we obtain a better node reduction than with any other setting. Among the (rer)-settings, `(rer)-0.1` is fastest regarding the solving time, but is still 6.7 % slower on average than the

Table 1: All instances

	98 instances solved by all				135 total	
	t (sec)	%	n	%	t (sec)	%
Settings						
`(fnt)-5`	81.9	100.0	3402.3	100.0	290.3	100.0
`(rer)-0.01`	86.9	106.1	2740.0	80.5	302.4	104.1
`(rer)-0.1`	85.5	104.4	3037.3	89.3	296.4	102.1
`(rer)-0.05`	86.7	105.9	2886.8	84.8	300.1	103.4
`(hyp)`	83.1	101.5	2804.4	82.4	290.5	100.1

Table 2: Large Trees: $n > 1000$ with at least one setting.

	53 instances solved by all				89 total	
	t (sec)	%	n	%	t (sec)	%
Settings						
`(fnt)-5`	216.1	100.0	46181.2	100.0	892.0	100.0
`(rer)-0.01`	236.1	109.2	33277.9	72.1	939.7	105.3
`(rer)-0.1`	230.5	106.6	38075.1	82.4	914.0	102.5
`(rer)-0.05`	236.2	109.3	35863.3	77.7	931.3	104.4
`(hyp)`	219.9	101.7	33502.6	72.5	888.6	99.6

reference setting.

The discrepancy between a reduction of the tree size and at the cost of more solving time per node is the result of a more aggressive use of strong-branching by the novel notions of reliability. The notion of (hyp)-reliability hereby appears to be more effective than relative-error reliability for guiding strong-branching effort because it focusses on resolving cases among the top pseudo-cost score branching candidates where the estimation alone may lead to inferior branching decisions.

For the sake of completeness, we also present the remaining instances, for which no solver took more than 1000 branch-and-bound nodes before termination, in Table 3. Out of the 46 instances in this group, there is only one, namely stp-3d, that could not be solved by any of the settings. All novel notions of reliability reduce the search tree size, although the effect is less striking than on the instances that required larger search trees. Note that the node reduction obtained with `(rer)-0.01` and even `(rer)-0.05` is now considerably better than the reduction obtained with (hyp)-reliability.

For those 37 instances for which optimality could not be proven within the time limit by at least one of our settings, we computed integrals of the dual-gap as a function of time. This measure, which was suggested in [ABH12, Ber13], attempts to compare the convergence of the dual gap towards zero. Table 4 shows the shifted geometric mean integral for all settings using a shift of 1000. All novel notions of reliability decrease the dual integral of the reference run, where the decrease is best with `(hyp)` yielding a reduction of more than 15 %. A similar result is obtained with `(rer)-0.1`, which outperforms other thresholds for (rer)-reliability in this respect.

6 Conclusions and future work

We introduced two novel notions of reliability: the (rer)-reliability based on pseudo-cost confidence intervals, and (hyp)-reliability implementing a variant of a 2-sample Student-t test. First experimental results with our implementation in SCIP show that these methods are promising for effectively reducing the size of branch-and-bound-trees compared to the current state-of-the-art fixed number threshold, especially for large trees. Our first implementation only considers pseudo-cost information, but can be readily applied to different history information such as, e.g., the inference history of a variable, as well.

In the computational study presented, we collected very little history information before using the statistical methods. Combining them with traditional fixed number threshold reliability might increase the power of the hypothesis and relative error thresholds significantly.

Note that for our computational experiments, we did not allow a dynamic adaption of the fixed number thresholds depending on the computational expenses on strong-branching during the search. Fixed number thresholds, however, show a superior performance if they are dynamically adjusted during the search, so that the overall strong-branching effort is kept reasonably small. For making a more effective use of the suggested approaches, it is necessary to let also the novel approaches dynamically adjust to problems for which strong-branching is very expensive. Note also that the variant of a 2-sample-t-test that we use for (hyp)-reliability is, in theory, only applicable when the two variables can be assumed to have equal variances. In practice, it would be possible to test for equal variances using an F-test and resort to the *Welch*-test if the variances are significantly unequal.

Acknowledgments

The author would like to thank Timo Berthold and Gerald Gamrath for their many valuable comments on earlier versions of this work. The work for this article has been conducted within the Research Campus Modal funded by the German Federal Ministry of Education and Research (fund number 05M14ZAM).

Table 3: Small trees: All solvers needed $n \leq 1000$ nodes.

	45 instances solved by all				46 total	
	t (sec)	%	n	%	t (sec)	%
Settings						
`(fnt)-5`	21.8	100.0	67.5	100.0	25.8	100.0
`(rer)-0.01`	22.3	102.3	55.9	82.9	26.3	102.2
`(rer)-0.1`	22.2	101.7	65.4	96.8	26.2	101.6
`(rer)-0.05`	22.2	101.7	59.4	88.0	26.2	101.6
`(hyp)`	22.1	101.4	62.4	92.5	26.1	101.3

Table 4: Shifted geom. mean dual integral for 37 time limit instances.

	$\Gamma^*(T)$	%
Settings		
`(fnt)-5`	73867.8	100.0
`(rer)-0.01`	71797.4	97.2
`(rer)-0.1`	62875.1	85.1
`(rer)-0.05`	70453.6	95.4
`(hyp)`	62747.0	84.9

References

[AB09] Tobias Achterberg and Timo Berthold. Hybrid branching. In Willem Jan van Hoeve and John N. Hooker, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 6th International Conference, CPAIOR 2009*, volume 5547 of *Lecture Notes in Computer Science*, pages 309–311. Springer, 2009.

[ABCC95] David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William J. Cook. Finding cuts in the TSP (A preliminary report). Technical Report 95-05, DIMACS, 1995.

[ABH12] Tobias Achterberg, Timo Berthold, and Gregor Hendel. Rounding and propagation heuristics for mixed integer programming. In Diethard Klatte, Hans-Jakob Lüthi, and Karl Schmedders, editors, *Operations Research Proceedings 2011*, pages 71–76. Springer Berlin Heidelberg, 2012.

[Ach07] Tobias Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2007.

[Ach09] Tobias Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[AKM04] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2004.

[AKM06] Tobias Achterberg, Thorsten Koch, and Alexander Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):1–12, 2006.

[BCMS98] Robert E. Bixby, Sebastiàn Ceria, Cassandra M. McZeal, and Martin W.P. Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, 58:12–15, 1998.

[Ber13] Timo Berthold. Measuring the impact of primal heuristics. *Operations Research Letters*, 41(6):611–614, 2013.

[BGG+71] Michel Bénichou, Jean-Michel Gauthier, Paul Girodet, Gerard Hentges, Gerard Ribière, and O. Vincent. Experiments in mixed-integer programming. *Mathematical Programming*, 1:76–94, 1971.

[BGS14] Timo Berthold, Gerald Gamrath, and Domenico Salvagnin. Cloud branching. Presentation slides from Mixed Integer Programming Workshop at Ohio State University. `https://mip2014.engineering.osu.edu/sites/mip2014.engineering.osu.edu/files/uploads/Berthold_MIP2014_Cloud.pdf`, 2014.

[CBC] CBC. COIN-OR branch-and-cut MIP solver. `https://projects.coin-or.org/Cbc`.

[CPL] CPLEX. IBM ILOG CPLEX Optimizer. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

[Dak65] R. J. Dakin. A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8(3):250–255, 1965.

[Dan08] Emilie Danna. Performance variability in mixed integer programming. Presentation slides from MIP workshop in New York City. `http://coral.ie.lehigh.edu/~jeff/mip-2008/program.pdf`, 2008.

[FM11] Matteo Fischetti and Michele Monaci. Backdoor Branching. In Oktay Günlück and Gerhard J. Woeginger, editors, *Integer Programming and Combinatorial Optimization*, volume 6655 of *Lecture Notes in Computer Science*, pages 183–191. Springer Berlin / Heidelberg, 2011.

[FM12] Matteo Fischetti and Michele Monaci. Branching on nonchimerical fractionalities. *OR Letters*, 40(3):159–164, 2012.

[Gam13] Gerald Gamrath. Improving strong branching by propagation. In Carla Gomes and Meinolf Sellmann, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 7874 of *Lecture Notes in Computer Science*, pages 347–354. Springer Berlin Heidelberg, 2013.

[GR77] J.-M. Gauthier and G. Ribière. Experiments in mixed-integer linear programming using pseudo-costs. *Mathematical Programming*, 12(1):26–47, 1977.

[GS11] Andrew Gilpin and Tuomas Sandholm. Information-theoretic approaches to branching in search. *Discrete Optimization*, 8(2):147 – 159, 2011.

[GUR] GUROBI. GUROBI Optimizer. `http://www.gurobi.com/products/gurobi-optimizer/gurobi-overview`.

[KAA+11] Thorsten Koch, Tobias Achterberg, Erling Andersen, Oliver Bastert, Timo Berthold, Robert E. Bixby, Emilie Danna, Gerald Gamrath, Ambros M. Gleixner, Stefan Heinz, Andrea Lodi, Hans Mittelmann, Ted Ralphs, Domenico Salvagnin, Daniel E. Steffy, and Kati Wolter. MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011.

[KKNS09] Fatma Kılınç Karzan, George L. Nemhauser, and Martin W. P. Savelsbergh. Information-based branching schemes for binary linear mixed integer problems. *Mathematical Programming Computation*, 1(4):249–293, 2009.

[LD60] A. H. Land and A. G Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[LS99] Jeff T. Linderoth and Martin W. P. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173–187, 1999.

[PC11] Jennifer Pryor and John W. Chinneck. Faster integer-feasibility in mixed-integer linear programs by branching to force change. *Computers & Operations Research*, 38(8):1143 – 1152, 2011.

[RF81] David M. Ryan and B. A. Foster. An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, pages 269–280. North Holland, Amsterdam, 1981.

[Rou14] George. G. Roussas. *A Course in Mathematical Statistics, Third Edition*. Elsevier Science & Technology Books, 2014.

[SCI] SCIP. SCIP. Solving Constraint Integer Programs. `http://scip.zib.de/`.

[SOP] SOPLEX. SoPlex. An open source LP solver implementing the revised simplex algorithm. `http://soplex.zib.de/`.

[XPR] XPRESS. FICO Xpress-Optimizer. `http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx`.

Appendix

This appendix contains an instance-wise outcome of our computational experiments described in Section 5. For each of the five settings, we present three columns; the measured dual integral $\Gamma^*(T)$, the number of nodes n, and the solving time in seconds t (sec). Table 5 shows the results for instances which we classified as small tree instances, and Table 6 contains the remaining instances, cf. Tables 3 and 2, respectively.

Table 5: Instance-wise experimental outcome for instances requiring at most 1000 nodes to solve.

Settings	(fnt)-5			(rer)-0.01			(rer)-0.1			(rer)-0.05			(hyp)		
	$\Gamma^*(T)$	n	t (sec)	$\Gamma^*(T)$	n	t (sec)	$\Gamma^*(T)$	n	t (sec)	$\Gamma^*(T)$	n	t (sec)	$\Gamma^*(T)$	n	t (sec)
Problem															
30n20b8	12756.5	18	196.3	14032.8	72	214.1	12854.5	11	198.6	12818.7	14	198.2	16010.0	97	239.4
air04	1862.4	8	37.9	1837.2	8	37.5	1827.1	8	37.3	1852.4	8	37.8	1726.3	8	35.3
air05	1270.2	62	25.6	1260.0	52	25.4	1285.2	74	25.8	1260.0	50	25.4	1295.5	94	26.1
app1-2	55296.9	41	875.6	108759.2	19	1736.2	108468.2	19	1731.6	108644.5	19	1734.5	50432.6	21	797.2
ash608gpia-3col	2000.0	7	20.0	2070.0	9	20.7	2070.0	9	20.7	2140.0	9	21.4	1990.0	7	19.9
blend2	23.4	240	0.6	34.2	126	0.8	24.2	275	0.7	28.8	190	0.7	18.8	220	0.6
dcmulti	0.3	8	0.8	0.4	14	1.0	5.5	14	1.2	0.3	14	0.8	0.5	12	1.0
fast0507	4444.4	630	147.8	1672.5	840	156.5	4273.6	588	140.2	975.8	648	147.6	6210.8	570	141.2
fiber	4.1	4	1.0	3.4	4	0.9	0.0	4	0.8	0.0	4	0.8	2.2	4	1.1
fixnet6	11.5	10	1.9	11.5	10	1.9	11.5	10	1.9	19.0	10	2.1	18.2	20	2.1
gesa2	5.0	3	0.4	0.1	3	0.6	5.0	3	0.3	5.0	3	0.4	5.0	3	0.5
gesa2-o	5.1	2	0.9	5.3	2	1.1	5.1	2	0.7	0.1	2	0.9	5.2	2	1.1
gesa3	5.1	7	1.0	5.1	7	1.2	5.2	9	1.3	5.1	9	1.0	5.2	9	1.2
gesa3_o	0.1	7	0.9	5.1	7	1.0	0.1	7	1.2	10.1	7	1.0	10.2	7	1.4
khb05250	0.2	4	0.5	0.2	4	0.5	0.2	4	0.5	0.2	4	0.5	0.2	4	0.5
l152lav	40.9	19	1.1	46.4	19	1.5	40.9	19	1.1	45.9	19	1.1	36.4	17	1.4
lseu	5.9	191	0.2	5.9	64	0.2	11.2	64	0.3	11.2	64	0.3	21.8	60	0.5
map18	15510.6	285	297.8	15805.5	275	303.3	16614.4	325	318.3	16787.0	325	321.6	15505.1	331	297.7
map20	12353.0	281	236.3	12752.0	307	243.8	12231.3	329	234.1	12561.2	307	240.3	11975.3	263	229.2
misc03	24.1	80	0.8	32.9	23	1.0	27.3	51	0.9	30.3	23	1.0	34.8	53	1.1
misc06	5.0	4	0.5	5.0	4	0.4	5.0	4	0.6	5.0	4	0.5	0.0	4	0.5
mod008	2.3	7	0.8	0.0	7	1.0	2.3	7	0.8	1.6	7	0.6	2.2	7	0.8
mod010	5.1	2	0.5	5.1	2	0.5	5.1	2	0.5	0.0	2	0.2	5.0	2	0.3
mod011	333.0	855	116.3	332.7	671	110.7	341.2	873	118.5	325.7	833	114.3	330.4	743	108.6
modglob	0.1	25	0.5	5.1	21	0.3	0.1	31	0.3	5.1	27	0.4	10.0	19	0.4
mspp16	98434.8	31	1856.2	93115.9	29	1755.9	92097.7	29	1736.7	92373.5	29	1741.9	132241.7	71	2493.7
mzzv42z	5458.9	110	155.5	5044.6	96	146.6	5070.4	210	147.2	5014.2	96	146.2	5135.8	155	148.0
neos-476283	1971.2	110	70.6	1976.2	107	67.9	1976.2	432	72.8	1971.2	379	71.6	1986.3	141	77.4
neos13	897.8	8	32.1	868.1	8	30.8	884.8	8	31.5	864.3	8	30.8	845.5	6	30.6
nw04	496.4	8	20.7	479.9	8	20.1	480.8	8	20.4	474.8	8	20.0	474.8	8	20.0
p0201	32.6	9	1.1	42.9	9	1.3	37.8	9	1.2	17.6	9	0.9	38.2	11	1.3
p0282	0.0	3	0.3	0.2	3	0.2	0.5	3	0.5	0.5	3	0.5	0.5	3	0.5
p2756	5.1	3	0.7	6.7	3	1.0	0.2	3	1.0	10.8	3	1.0	1.7	3	0.9
pp08a	21.2	161	0.7	26.5	51	0.9	26.2	51	0.8	31.5	51	1.0	18.4	57	1.2
pp08aCUTS	1.5	153	0.8	21.5	51	1.0	17.0	49	1.1	21.7	51	1.1	3.1	59	1.4
qnet1	10.2	3	2.0	12.0	3	2.1	9.8	3	1.8	9.6	3	1.8	10.5	3	2.0
qnet1_o	0.0	4	1.3	0.0	4	1.5	0.0	4	1.1	2.1	4	1.2	0.0	4	1.3
rail507	529.0	644	147.1	684.5	546	137.3	570.5	530	137.0	884.9	612	145.2	791.6	488	135.5
rentacar	80.6	4	3.4	91.3	4	3.6	66.6	4	3.2	71.6	4	3.3	91.2	4	3.6
rmatr100-p10	6821.8	709	120.1	7118.5	793	125.3	7027.2	791	123.7	7005.8	793	123.4	7118.4	731	125.3
rmatr100-p5	14256.9	349	235.6	15283.4	367	252.5	15198.3	373	251.1	15368.5	337	253.9	14396.6	319	237.9
set1ch	0.0	3	0.6	0.0	3	0.6	1.2	3	0.5	0.0	3	0.7	1.2	3	0.6
stp3d	145433.0	14	7200.0	145530.6	14	7200.0	145823.5	13	7200.0	145530.6	14	7200.0	145791.6	17	7200.0
tanglegram1	99072.9	33	991.9	77887.9	27	779.8	78127.7	29	782.2	78607.1	27	787.0	90433.1	31	905.4
tanglegram2	793.1	3	8.0	822.8	3	8.3	842.7	3	8.5	783.1	3	7.9	793.1	3	8.0
vpm2	19.6	272	1.0	24.8	50	1.1	24.8	186	1.1	24.2	72	1.0	14.3	160	0.9

Table 6: Instance-wise experimental outcome for instances for which one setting required more 1000 nodes.

Settings	(fnt)-5			(rer)-0.01			(rer)-0.1			(rer)-0.05			(hyp)		
	$\Gamma^*(T)$	n	t (sec)	$\Gamma^*(T)$	n	t (sec)	$\Gamma^*(T)$	n	t (sec)	$\Gamma^*(T)$	n	t (sec)	$\Gamma^*(T)$	n	t (sec)
Problem															
a1c1s1	259863.8	2070335	7200.0	259882.0	1294388	7200.0	259875.0	2081884	7200.0	259875.3	1668708	7200.0	259875.2	1240319	7200.0
aflow30a	377.5	1246	10.1	511.8	1908	12.4	503.4	1758	12.5	498.1	1690	12.4	486.4	1366	12.0
aflow40b	16390.8	143682	536.0	32073.3	183576	905.9	29016.8	148296	819.2	29786.2	167692	872.2	23662.9	119722	659.3
arki001	105.7	365939	7200.0	110.7	616240	7200.0	105.7	407503	7200.0	90.7	325090	7200.0	105.7	778086	7200.0
atlanta-ip	70469.7	8529	7200.0	70460.7	10011	7200.0	70455.8	12637	7200.0	70465.1	10872	7200.0	70456.2	8181	7200.0
bab5	6632.3	96735	7200.0	6624.1	110967	7200.0	6644.8	113586	7200.0	6633.1	105996	7200.0	6706.7	110285	7200.0
beasleyC3	92907.7	2841834	7200.0	92897.0	1807660	7200.0	92900.9	2136808	7200.0	92901.6	1465523	7200.0	94245.3	1511264	7200.0
bell3a	16.3	22611	2.8	-4.0	20027	2.9	6.4	21353	3.0	1.3	20261	3.1	1.2	20047	3.3
bell5	10.0	1152	0.3	10.0	956	0.2	10.0	968	0.3	10.0	956	0.3	10.0	1025	0.3
biella1	33983.5	6883	678.8	35540.9	8195	709.9	24774.0	5013	494.9	29386.2	6903	587.0	46142.6	9683	921.6
bienst2	29985.7	309529	561.7	10698.3	101007	427.5	15980.8	113088	463.9	10689.9	101007	427.2	12885.1	79381	336.6
binkar10_1	175.9	142580	139.1	625.5	107739	162.9	175.9	101180	159.9	2049.7	106491	163.5	510.4	85524	145.7
cap6000	30.0	1801	1.5	25.0	738	1.6	20.0	738	1.6	15.0	738	1.5	25.0	1031	1.9
cov1075	265179.6	2274377	4821.6	259198.5	2122029	4712.7	251729.5	2162377	4576.9	267102.0	2248711	4856.4	247810.6	1743871	4505.8
csched010	84082.4	1035664	7200.0	84128.2	603676	7200.0	84015.8	568709	7200.0	84075.1	572630	7200.0	127721.8	455625	6224.6
danoint	20001.8	1102583	4476.0	32307.8	1083111	7200.0	30548.3	1611094	6832.5	32248.8	1635143	7200.0	31991.8	1310682	7127.9
dfn-gwin-UUM	1333.5	68085	93.1	1080.5	54829	134.2	2312.0	57135	151.2	1090.5	54813	136.3	2415.4	53901	133.3
ds	273445.3	7413	7200.0	273436.1	8487	7200.0	274261.5	9598	7200.0	273432.0	6367	7200.0	273466.0	6133	7200.0
eil33-2	1351.2	340	30.3	1160.5	1330	38.8	717.9	1530	34.9	732.5	1488	39.2	1865.1	1282	35.7
eilB101	12660.3	6284	265.1	3198.6	5724	237.1	14742.9	8236	300.0	12306.0	6848	255.5	12453.5	6956	241.9
enlight13	13838.2	522384	158.9	1247.5	19169	13.6	4636.2	100856	49.3	13410.4	307626	143.0	4738.9	95831	49.5
glass4	240002.3	12763342	7200.0	52685.8	3004703	2525.4	15976.8	1492813	950.2	17036.8	1312334	1013.7	9243.4	853465	544.7
gmu-35-40	60.0	32198403	7200.0	65.0	19990677	7200.0	65.0	20646471	7200.0	65.0	20679476	7200.0	65.0	22444647	7200.0
harp2	233.9	448438	160.7	65.2	200954	142.5	440.4	211217	146.0	384.4	240647	158.7	43.7	160188	106.1
iis-100-0-cov	24816.6	76155	523.3	26388.9	63349	691.4	24227.7	65763	563.8	30571.0	68587	739.9	21187.0	71295	571.7
iis-bupa-cov	58125.2	181441	2500.9	125061.4	146447	3239.1	58533.6	180635	2555.8	64676.0	151591	2824.4	61793.2	152075	2590.1
iis-pima-cov	13735.8	5869	248.4	13223.7	5821	239.6	15055.9	6829	270.8	14064.2	6231	253.9	17366.9	6911	307.7
macrophage	294851.0	2875395	7200.0	294856.2	1758872	7200.0	294850.5	2618474	7200.0	294854.0	2331049	7200.0	294856.6	1671290	7200.0
markshare1	720000.0	102171542	7200.0	720000.0	101365477	7200.0	720000.0	100587986	7200.0	720000.0	101315675	7200.0	720000.0	99735165	7200.0
markshare2	720000.0	88721939	7200.0	720000.0	86287896	7200.0	720000.0	87139578	7200.0	720000.0	87388660	7200.0	720000.0	88582090	7200.0
mas74	3306.0	2752472	342.5	3875.7	2112420	404.1	2801.8	2191551	294.1	3391.2	2476385	352.7	4110.5	2147147	424.3
mas76	244.8	547767	51.4	143.4	381797	55.2	257.9	412299	46.0	307.1	394475	54.2	326.2	175794	28.3
mcsched	26190.3	46543	603.9	65737.5	108389	1334.1	40368.9	67915	862.8	55540.1	89133	1145.4	88612.8	135383	1760.7
mik-250-1-100-1	5995.2	6996847	1372.3	2386.9	1481627	565.6	1567.0	1490147	367.1	1531.9	773126	340.1	1464.3	814873	336.7
mine-166-5	1250.2	2038	23.1	1333.9	328	24.5	1403.7	366	25.8	1355.8	328	25.0	1728.8	1268	31.1
mine-90-10	20332.3	197329	428.7	10188.4	75221	244.1	12635.2	94094	293.4	12630.8	95405	293.9	3224.5	31458	106.5
misc07	1201.8	47624	26.3	1172.1	32697	27.5	1888.5	28048	25.3	1135.2	32087	26.7	2072.6	33089	27.8
mkc	5504.2	2175182	7200.0	5493.4	1388742	7200.0	5498.5	1988218	7200.0	5493.1	2248552	7200.0	5504.3	1923042	7200.0
momentum1	85140.1	519796	7200.0	85139.7	394564	7200.0	85138.6	231117	7200.0	85138.3	225837	7200.0	85137.9	186601	7200.0
momentum2	226569.4	183471	4366.0	202624.3	237101	7200.0	232489.8	107062	4144.7	202624.1	206582	7200.0	202628.3	150449	7200.0
msc98-ip	5191.4	58884	7200.0	5196.4	9653	7200.0	5191.4	103812	7200.0	5191.3	16542	7200.0	5191.4	15535	7200.0
mzzv11	10185.3	2014	324.5	9704.5	1857	317.8	9566.5	1939	316.2	9638.8	1857	316.8	10181.3	1504	325.4
n3div36	52928.9	513914	7200.0	54503.2	321848	7200.0	54508.1	363298	7200.0	54509.2	349172	7200.0	54518.7	339407	7200.0
n3seq24	4815.7	8601	7200.0	4805.7	10461	7200.0	4825.7	9340	7200.0	4815.7	10104	7200.0	4820.7	9912	7200.0
n4-3	9826.5	46719	557.0	47258.8	44277	855.3	13487.6	43883	903.8	8737.2	41077	802.0	42477.9	39805	769.4
neos-1109824	1674.8	17347	84.2	2735.5	7488	50.9	409.6	10240	59.4	3544.4	11154	65.8	4781.5	16561	88.6
neos-1337307	2841.1	714364	7200.0	2841.1	501390	7200.0	2846.2	458694	7200.0	2856.3	476398	7200.0	2856.4	484765	7200.0
neos-1396125	37201.4	88307	836.4	113591.1	58386	2698.6	69995.0	64241	1819.8	90164.5	56132	2120.4	31976.2	50484	897.9
neos-686190	2705.3	1865	44.5	2672.3	2263	43.9	2678.4	2263	44.0	2707.6	2263	44.6	2653.9	2135	43.6
neos-916792	117042.3	2065470	7200.0	115048.4	1119717	7200.0	119270.2	2018719	7200.0	117825.1	1857375	7200.0	124350.3	1258952	7200.0
neos18	2054.8	59329	79.2	4839.4	93832	175.7	3193.1	96694	125.7	2389.0	65632	103.9	5294.7	78624	158.0
net12	231010.4	3139	2842.7	289377.8	3642	3560.4	209691.2	2471	2580.6	201824.1	2689	2483.8	248412.3	3519	3056.7
netdiversion	59752.9	2430	7200.8	59557.6	2439	7200.5	59410.5	2442	7200.3	59949.6	2441	7200.8	59459.5	1215	7200.3
newdano	166449.7	3194194	3936.2	201008.7	2050673	4502.9	120312.9	2069380	2634.5	159163.1	2032092	3606.1	134323.4	1817997	3238.2
noswot	16508.7	1607445	315.5	8445.3	579588	161.4	11830.8	1197750	226.1	7958.7	564187	152.1	9334.9	603130	178.4
ns1208400	49760.0	881	497.6	52960.0	860	529.6	52940.0	860	529.4	52900.0	860	529.0	49460.0	1448	494.6
ns1688347	690.6	1088	12.4	702.0	311	12.6	748.0	877	13.4	770.9	1215	13.8	598.7	184	10.8
ns1766074	237360.0	848597	2373.6	355190.0	750095	3551.9	269760.0	774224	2697.6	340590.0	718315	3405.9	372800.0	741809	3728.0
ns1830653	9767.1	22244	219.9	30774.9	44597	429.1	27173.7	44991	379.0	30621.1	44597	427.0	15492.2	37047	319.8
nsrand-ipx	10543.7	3978440	6502.4	11113.6	2418018	7200.0	11109.1	5071964	7200.0	14255.2	4148300	6990.6	11560.5	2917724	7200.0
opm2-z7-s2	46319.1	1283	770.8	45760.2	1421	761.5	48536.5	1469	807.7	47454.8	1461	789.7	54671.9	1569	909.8
pg5_34	1535.8	136710	713.1	35581.2	94226	828.6	1910.0	64430	757.4	33633.6	71942	794.7	4927.1	69658	787.6
pigeon-10	72000.0	3010254	7200.0	72000.0	2360046	7200.0	72000.0	3252076	7200.0	72000.0	3278150	7200.0	72000.0	10018235	7200.0
pk1	7282.0	393967	74.0	8180.0	255785	81.8	6175.0	341483	74.5	8180.0	314013	81.8	9759.2	305553	99.2
protfold	169388.3	6014	7200.0	169376.0	3610	7200.0	169391.8	3651	7200.0	169372.2	4933	7200.0	169372.2	4653	7200.0
pw-myciel4	432085.2	3293759	7200.0	432083.2	2898912	7200.0	432079.2	1813613	7200.0	432080.8	4477059	7200.0	432080.8	1035796	7200.0
qiu	4489.9	9557	48.7	5801.3	8355	68.7	6397.8	8383	69.4	5790.1	8355	68.6	6017.2	8477	71.0
ran16x16	1222.1	284383	177.9	1838.2	255537	249.9	1614.4	242583	247.3	1670.8	261092	257.7	2517.7	298805	304.6
rd-rplusc-21	719565.9	563331	7200.0	719565.9	179321	7200.0	719565.9	239970	7200.0	719565.9	334989	7200.0	719565.9	413155	7200.0
reblock67	789.7	49872	75.6	644.9	34120	73.7	576.9	32718	77.0	641.6	33970	74.0	2556.8	48376	90.3
rmine6	28254.0	367462	608.8	1152.5	273034	811.0	36713.5	499955	787.2	29021.5	319661	639.5	643.2	260116	712.0
rocII-4-11	219386.9	140524	6949.1	182328.2	11390	2765.6	194016.7	43812	5435.6	131867.9	21426	3746.1	93852.9	6597	1428.5
rococoC10-001000	78134.2	2225139	7200.0	78139.8	1039332	7200.0	78123.2	1282843	7200.0	78140.9	1222227	7200.0	78127.7	960482	7200.0
roll3000	30528.9	2722022	7200.0	30535.5	1817076	7200.0	30528.6	1690055	7200.0	30528.0	1596804	7200.0	30534.2	1756241	7200.0
rout	528.1	19618	22.0	1259.0	24804	39.0	457.0	41362	48.7	410.5	30728	44.0	1331.4	25720	40.0
satellites1-25	149555.0	5749	1709.2	215223.8	6168	2459.7	215346.2	6168	2461.1	215923.8	6168	2467.7	205954.9	11257	2731.7
seymour	22808.8	438631	7200.0	22808.8	265347	7200.0	22809.0	368251	7200.0	22818.7	304376	7200.0	22818.6	256077	7200.0
sp97ar	715182.9	2862	7200.1	715183.0	2861	7200.1	715183.0	2860	7200.0	715182.9	2864	7200.1	715182.9	2862	7200.1
sp98ic	642833.8	33432	7200.0	644543.1	24268	7200.1	645061.6	22414	7200.1	642711.1	23907	7200.0	644419.0	22865	7200.1
sp98ir	1661.9	2582	37.1	1550.4	2928	34.8	1571.2	2898	35.6	1570.6	3075	35.2	1651.7	2972	36.8
stein27	38.3	4073	0.6	108.6	933	1.7	83.1	1471	1.3	95.8	903	1.5	57.5	2255	0.9
stein45	430.7	49451	6.8	263.7	41231	10.4	658.7	48083	10.4	690.3	41669	10.9	677.7	34893	10.7
swath	131547.9	1526206	7200.0	131194.0	1063757	7200.0	132446.5	772490	7200.0	132446.7	1029791	7200.0	132805.6	939871	7200.0
timtab1	10270.7	826333	292.0	17020.1	945269	517.0	15603.8	785223	451.0	14122.2	670753	401.2	10924.0	690625	396.1
timtab2	323566.6	13961074	7200.0	323565.2	8315730	7200.0	323563.5	11616959	7200.0	323567.6	8763724	7200.0	323566.2	6315656	7200.0
tr12-30	437.4	1072845	1082.6	710.8	1362243	2120.5	817.2	1783399	2388.3	2355.5	1278417	2070.5	719.9	814811	1264.4
unitcal_7	205323.5	112703	4696.1	216041.2	78003	5031.6	3766.4	116463	7200.0	250737.6	92119	5725.5	3791.0	181281	7200.0
vpphard	720000.0	58045	7200.0	720000.0	45240	7200.0	720000.0	29866	7200.0	720000.0	31949	7200.0	720000.0	12657	7200.0
zib54-UUE	66689.8	347875	2336.9	91141.3	294299	3270.2	104695.5	544214	3835.0	95583.3	288932	3337.3	79035.2	278909	3218.6