GREGOR HENDEL[1]

Adaptive Large Neighborhood Search for Mixed Integer Programming

[1] 0000-0001-7132-5142

ZIB Report 18-60 (December 2018)

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30-84185-0
Telefax: +49 30-84185-125

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

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

Adaptive Large Neighborhood Search for Mixed Integer Programming

Gregor Hendel*

December 18, 2018

Abstract

Large Neighborhood Search (LNS) heuristics are among the most powerful but also most expensive heuristics for mixed integer programs (MIP). Ideally, a solver learns adaptively which LNS heuristics work best for the MIP problem at hand in order to concentrate its limited computational budget.

To this end, this work introduces Adaptive Large Neighborhood Search (ALNS) for MIP, a primal heuristic that acts a framework for eight popular LNS heuristics such as Local Branching and Relaxation Induced Neighborhood Search (RINS). We distinguish the available LNS heuristics by their individual search domains, which we call *neighborhoods*. The decision which neighborhood should be executed is guided by selection strategies for the *multi armed bandit problem*, a related optimization problem during which suitable actions have to be chosen to maximize a reward function. In this paper, we propose an LNS-specific reward function to learn to distinguish between the available neighborhoods based on successful calls and failures. A second, algorithmic enhancement is a generic variable fixing priorization, which ALNS employs to adjust the subproblem complexity as needed. This is particularly useful for some neighborhoods which do not fix variables by themselves. The proposed primal heuristic has been implemented within the MIP solver SCIP. An extensive computational study is conducted to compare different LNS strategies within our ALNS framework on a large set of publicly available MIP instances from the MIPLIB and Coral benchmark sets. The results of this simulation are used to calibrate the parameters of the bandit selection strategies. A second computational experiment shows the computational benefits of the proposed ALNS framework within the MIP solver SCIP.

1 Introduction

Mixed integer programming (MIP) is a powerful modeling paradigm with numerous relevant industrial applications in scheduling, production planning, traffic optimization [8] and countless more. For solving their models, many practitioners rely on state-of-the-art commercial or noncommercial general purpose MIP solvers such as CPLEX [12], XPress [35], SCIP [1, 22], or CBC [10], all of which employ a variant of the branch-and-bound algorithm [13, 29]. As the branch-and-bound algorithm by itself may be quite slow to provide good solutions, a rich set of *primal heuristic algorithms* has been proposed to improve the primal convergence [5] of the solvers. Primal heuristics can be further classified [2] into rounding algorithms, diving and objective

*Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, `hendel@zib.de`

diving heuristics and feasibility-pump [2, 18] procedures, and finally Large Neighborhood Search (LNS) heuristics such as *Relaxation Induced Neighborhood Search* (RINS) [14]. LNS heuristics typically restrict the search space of an input MIP instance to a particular neighborhood of interest. The resulting *auxiliary problem* (cf. Definition 1) is again a MIP, which is then partially solved by a branch-and-bound algorithm under reasonable working limits, and eventual solutions are kept for the main search process. Many different LNS techniques have been proposed in recent years [15, 14, 33, 21, 6, 17]. Their computational effort makes it impractical to apply all of them frequently within the solver. Since a priori, it is unclear which approach might work best for a concrete problem instance, the solver ideally learns during the solving process which LNS heuristics should be applied, and more importantly, which ones can be deactivated.

Following this line of thought, we propose Adaptive Large Neighborhood Search (ALNS) for MIP. We address in particular the question how to select from the set of available neighborhoods, which are introduced in Section 2. In Section 3, we propose a suitable reward function for LNS heuristics to learn to discriminate between the neighborhoods during the search. We also propose a generic variable fixing scheme that can be used to extend the set of fixed variables within a selected neighborhood to reach a target fixing rate. This has a particular impact on LNS heuristics that do not fix variables by themselves and may hence be too expensive on larger problems, such as, e.g., Local Branching [15] (cf. Section 2.2).

The framework is obliged to trade off between exploration and exploitation, because only one neighborhood is selected and evaluated at a specific call. Such a selection scenario is also referred to as *multi armed bandit problem*, in which a player tries to maximize their reward by playing one available action at a time and observing the particular reward of this action only. We review three selection algorithms for the multi-armed bandit problem in Section 4. Two numerical experiments are presented in Section 5, a first one to tune the selection process of the ALNS heuristic, and a second experiment to show that ALNS improves the MIP performance of SCIP on a large set of publicly available benchmark instances from the collections of MIPLIB [7, 3, 27] and Cor@l [11].

1.1 Related Work

The notion of an Adaptive Large Neighborhood Search has already been coined in the literature, particularly in the context of Constraint Programming, where ALNS is usually tailored to a particular application. The authors [28] were the first to describe an adaptive LNS technique for single-mode scheduling problems, which selects from a finite set of so-called search operators, which are a CP analogue to the neighborhoods for general MIP (see Section 2). Building upon their method, ALNS has also been applied for different types of Vehicle Routing Problems, see [32] for an overview. Throughout the remainder of this article, we will shortly write ”ALNS” to denote our proposed ”ALNS for MIP”.

A different, MIP specific approach to learn how to run heuristics has been recently proposed in [26]. Their work uses logistic regression to predict the probability of success for different diving heuristics. The prediction is based on state information about the current node and the overall search. Their approach is fundamentally different from our proposed method in that it learns one regression for each individual diving heuristic, but does not attempt to prioritize between them.

The first use of bandit related ideas inside MIP solvers [34] concerns the integration of a node selection rule into CPLEX. This node selection approach balances exploration and exploitation of the solution process inspired by a successful method for game search trees, which is related to the Upper Confidence Bounds selection algorithm [9], which is explained in Section 4.1.

2 Large Neighborhood Search Heuristics for MIP

We propose Adaptive Large Neighborhood Search in the context of mixed-integer programs (MIPs). A MIP P is an optimization problem of the form

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

in n variables and m linear constraints, which are defined by a matrix $A \in \mathbb{Q}^{m,n}$ and a *right hand side* $b \in \mathbb{Q}^m$. Every variable x_j, $j \in \{1,\dots,n\}$ has an *objective coefficient* $c_j \in \mathbb{Q}$ and *a lower and upper bound* denoted by $l_j \in \mathbb{Q} \cup \{-\infty\}$ and $u_j \in \mathbb{Q} \cup \{+\infty\}$, respectively. Finally, without loss of generality, the first n_i variables are further constrained that they must take integer solution values. These are called the *integer variables* of P. An important subset of the integer variables are the $n_b \leq n_i$ *binary variables* with a $\{0,1\}$-domain. The shorthand notations for binary variables and nonbinary integer variables, which are called *general integer variables*, are $N_b = \{1,\dots,n_b\}$ and $N_i = \{n_b + 1,\dots,n_i\}$, respectively. Binary variables are often used to model highly relevant yes/no-decisions in an optimization scenario such as, e.g., if a facility should be built at a certain location. Therefore, binary variables often receive a prioritized treatment by the neighborhoods in Sections 2.1 and 2.2. Every point of $\mathbb{Q}^n$ satisfying all of the above constraints is called a *solution* of P, and the set of all solutions of P is denoted by $\mathscr{S}_P$.

Dropping all integrality restrictions yields the *LP relaxation* of P. It is well known that (MIP) in the presented general form is $\mathscr{N}\mathscr{P}$-hard to solve, which is why all modern MIP solvers employ some form of *branch-and-bound* algorithm [29, 13]. In essence a clever enumeration, the branch-and-bound algorithm repeatedly partitions the search space of an input MIP P, mainly guided by integer variables with noninteger (fractional) values in the solution x^{lp} to its LP relaxation. Since the LP relaxation has fewer constraints than P and hence a broader feasible region, its optimal objective value $c^T x^{\text{lp}}$ is a lower bound to the *optimal value* c^* of P. If, in addition, the LP solution satisfies the integrality requirements $x_j^{\text{lp}} \in \mathbb{Z}\; \forall j \in \{1,\dots,n_i\}$, then $c^T x^{\text{lp}} = c^*$ and x^{lp} is an optimal solution for P. The minimum lower bound of all unprocessed subproblems is called the *dual bound* and denoted by c^{dual}.

In practice, however, the LP relaxation mostly provides feasible solutions only at deeper levels, ie. later stages of the search tree. Many different primal heuristic algorithms have been proposed to overcome this weakness, which are highly diverse in the computational effort they require. Starting from simple and fast heuristics [24] that attempt to construct feasible solutions by rounding the LP solution, a higher computational effort is usually required by diving or feasibility-pump [18] like procedures, which solve modified LP relaxations. At the most expensive end of the scale lies the class of *Large Neighborhood Search* (LNS) heuristics that solve an *auxiliary problem* with branch-and-bound techniques.

Definition 1 (Auxiliary problem) *Let P be a MIP with n variables. For a polyhedron $\mathscr{N} \subseteq \mathbb{Q}^n$ and objective coefficients $c_{aux} \in \mathbb{Q}^n$, a MIP P^{aux} defined as*

$$\min\left\{c_{aux}^T x \,|\, x \in \mathscr{S}_P \cap \mathscr{N}\right\} \tag{1}$$

is called an auxiliary problem *of P. The polyhedron $\mathscr{N}$ is called* neighborhood.

In other words, P^{aux} has the same number of variables (columns) as the original MIP P. Its solution set is a subset of $\mathscr{S}_P$. Hence, every solution to P^{aux} is a solution for P. Definition 1 requires $\mathscr{N}$ to be a polyhedron, ie., it should be expressed by a finite set of inequalities. The definition includes $\mathscr{N} = \mathbb{Q}^n$. Each neighborhood has its associated auxiliary objective function, which can be different from the main objective function of P.

There are only a few different types of neighborhoods typically used. All LNS heuristics have in common that they solve auxiliary problems around a set of reference points to either provide a first solution or, as in most cases, an improvement to the current incumbent solution. One of the most common classes of neighborhoods is derived by considering a set of reference points and fixing integer variables whose values are integer and agree on all those points.

Definition 2 (Fixing neighborhood) *Let P be a MIP with n variables and $n_i \leq n$ integer variables. Let $\mathscr{M} \subseteq \{1,\dots,n_i\}$ and $x^* \in \mathbb{Q}^n$. A* fixing neighborhood

$$\mathscr{N}^{fix}(\mathscr{M},x^*) := \left\{x \in \mathbb{Q}^n \,|\, x_j = x_j^* \;\forall j \in \mathscr{M}\right\}$$

fixes the subset $\mathscr{M}$ to their values in x^*.

Definition 3 (Matching set) *For $k \geq 1$, let $X = \{x^1,\dots,x^k\} \subset \mathbb{Q}^n$ with $x^i \neq x^{i'} \;\forall i \neq i' \in \{1,\dots,k\}$. The* matching set

$$\mathscr{M}^{=}(X) := \left\{j \in \{1,\dots,n_i\} \,|\, x_j^i = x_j^1 \;\forall i \in \{1,\dots,k\}\right\}$$

describes all integer variable indices whose values agree on X. We call X the set of reference points.

Combining Definitions 2 and 3 is very popular for constructing auxiliary problems. Starting from a set of reference points $X = \{x^1,\dots,x^k\}$, a fixing neighborhood is obtained with the help of the matching set of X. Since all points in X agree on their matching set $\mathscr{M}^{=}(X)$, the same fixing neighborhood is obtained regardless of the anchor point

$$\mathscr{N}^{\text{fix}}\left(\mathscr{M}^{=}(X),x^1\right) = \mathscr{N}^{\text{fix}}\left(\mathscr{M}^{=}(X),x^i\right) \quad \forall i \in \{1,\dots,k\}.$$

It should be noted that whenever a set of reference points X contains at least one solution $x \in \mathscr{S}_P$, the auxiliary MIP defined by the fixing neighborhood of the matching set is feasible because $x \in \mathscr{S}_P \cap \mathscr{N}^{\text{fix}}(\mathscr{M}^{=}(X),x)$.

The task of finding an improving solution can be easily incorporated into Definition 1. Assume that an incumbent solution $x^{\text{inc}} \in \mathscr{S}_P$ and a dual bound c^{dual} are already available, and let a given neighborhood $\mathscr{N} \subseteq \mathbb{Q}^n$ contain some nonimproving solutions. For $\delta \in (0,1)$, every solution $x \in \mathscr{S}_P$ that satisfies

$$c^T x \leq (1-\delta) \cdot c^T x^{\text{inc}} + \delta \cdot \underbrace{c^{\text{dual}}}_{< c^T x^{\text{inc}}} < c^T x^{\text{inc}}$$

is clearly an improving solution. The set of solutions that are better than x^{inc} by at least δ is contained in the *improvement neighborhood*

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

Therefore, refining the original neighborhood as

$$\mathscr{N}' = \mathscr{N} \cap \mathscr{N}^{\text{obj}}\left(\delta, x^{\text{inc}}\right)$$

filters out all nonimproving solutions, regardless of the choice of c_{aux}. The choice of δ is an important control parameter to weigh between the difficulty (and feasibility) of the sub-MIP and the desired amount of improvement. The above neighborhood notions suffice to describe several popular LNS heuristics.

2.1 Fixing Neighborhood LNS Heuristics

RINS [14] *Relaxation Induced Neighborhood Search* (RINS) is one of the first LNS approaches. The idea of RINS is to fix integer variables whose solution values agree in the solution x^{lp} of the LP relaxation at the current, local node, and the current incumbent solution x^{inc}. After the choice of a suitable improvement δ, the neighborhood of the auxiliary MIP of RINS is

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

Crossover [33] Another improvement heuristic is the *Crossover* heuristic, which is inspired by the recombination of solutions within genetic algorithms. Crossover selects $k \geq 2$ already known, feasible solutions $X = \{x^1, \dots, x^k\} \subseteq \mathscr{S}_P$ as reference points. X does not necessarily contain x^{inc}. The *crossover neighborhood* fixes

$$\mathscr{N}_{\text{Cross}} := \mathscr{N}^{\text{fix}}\left(\mathscr{M}^{=}(X), x^1\right) \cap \mathscr{N}^{\text{obj}}(\delta, x^{\text{inc}}).$$

The authors suggest to use $k = 2$ solutions that are randomly selected from all available solutions, using a bias towards solutions with better objective.

Mutation [33] Furthermore, the authors suggest a second LNS heuristic called *Mutation* that fixes a random subset of integer variables of the incumbent solution. For a randomly chosen subset $\mathscr{M}^{\text{rand}} \subseteq N_b \cup N_i$, the *mutation neighborhood* is defined as

$$\mathscr{N}_{\text{Muta}} := \mathscr{N}^{\text{fix}}\left(\mathscr{M}^{\text{rand}}, x^{\text{inc}}\right) \cap \mathscr{N}^{\text{obj}}\left(\delta, x^{\text{inc}}\right).$$

In order to control the difficulty of the sub-MIP, the obvious input to the mutation neighborhood is a number or percentage of integer variables that should be fixed. Mutation is the only neighborhood for which this number can be controlled directly: all previous neighborhoods depend on the cardinality of their matching set.

RENS [6] Starting from an LP relaxation solution x^{lp}, the *Relaxation Enforced Neighborhood Search* (RENS) neighborhood focusses on the feasible roundings of x^{lp} and can be written as

$$\mathscr{N}_{\text{RENS}} := \left\{ x \in \mathbb{Q}^n \mid \left\lfloor x_j^{\text{lp}} \right\rfloor \leq x_j \leq \left\lceil x_j^{\text{lp}} \right\rceil, j \in \{1, \ldots, n_i\} \right\}.$$

Similarly to the RINS heuristic, the aim of RENS is to construct feasible solutions that are close to the LP relaxation solution and therefore have a near-optimal solution value.

2.2 LNS Heuristics Using Constraints and Auxiliary Objective Functions

All approaches presented so far fix a set of integer variables using one or several reference points. *Local Branching* [15] is the first LNS heuristic that uses a different neighborhood.

Local Branching [15] Instead of fixing a set of variables and solving for improving solution values on the remaining variables, the neighborhood of Local Branching is restricted to a ball around the current incumbent solution. More formally, Let P be a MIP with $n_b \geq 1$ binary variables. Based on the L_1-norm or Manhattan metric for $x \in \mathbb{Q}^n$, the *binary norm*[1] of x is defined as

$$\|x\|_b := \sum_{j=1}^{n_b} |x_j|.$$

Let $x^{\text{inc}} \in \mathscr{S}_P$ be an incumbent solution for P, and let $d_{\max} > 0$ denote a distance cutoff parameter. The *local branching neighborhood* is the restriction

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

The reason for preferring the binary norm over the regular norm or the norm taking all integer variables is practicality. The binary norm can be expressed as a linear constraint without introducing auxiliary variables.

Proximity Search [17] A dual version of Local Branching has been introduced as *Proximity Search*. Using the binary norm, Proximity seeks to minimize the binary norm $\left\| x - x^{\text{inc}} \right\|_b$ through an auxiliary objective coefficient vector $c_{\text{Proxi}} = c_{\text{Proxi}}(x^{\text{inc}})$ defined as

$$(c_{\text{Proxi}})_j := \begin{cases} 0 & \text{if } j > n_b \\ 1 & \text{if } x_j^{\text{inc}} = 0 \\ -1 & \text{if } x_j^{\text{inc}} = 1 \end{cases}$$

over the entire set of improving solutions:

$$\mathscr{N}_{\text{Proxi}} := \mathscr{N}^{\text{obj}}(\delta, x^{\text{inc}}).$$

[1]Strictly speaking, $\|\cdot\|_b$ is a seminorm because nonzero vectors can have binary norm of 0.

Zero Objective The second LNS heuristic that uses an auxiliary objective function different from the original objective function is *Zero Objective*. If an incumbent solution x^{inc} is available, Zero Objective searches the entire set of improving solutions $\mathcal{N}_{\text{Zeroobj}} := \mathcal{N}^{\text{obj}}(\delta, x^{\text{inc}})$ but uses $c_{\text{Zeroobj}} := 0$ as auxiliary objective function. If no incumbent is available, $\mathcal{N}_{\text{Zeroobj}} = \mathbb{Q}^n$. Zero Objective thereby reduces the search for an (improving) solution to a feasibility problem.

DINS [21] Distance Induced Neighborhood Search (DINS) combines elements of the Crossover, Local Branching, and RINS heuristics. Similarly to RINS, the intuition is that improving solutions are located between the current incumbent solution x^{inc} and the solution to the node LP relaxation x^{lp}. With the intention of reducing the *integer distance*

$$\left\| x^{\text{inc}} - x^{\text{lp}} \right\|_i := \sum_{j=1}^{n_i} |x_j^{\text{inc}} - x_j^{\text{lp}}|,$$

let $J := \{j \mid |x_j^{\text{inc}} - x_j^{\text{lp}}| \geq 0.5\} \subseteq N_i$ denote the index set of general integer variables with a difference of at least 0.5 between the two reference points. The J-neighborhood of DINS is

$$\mathcal{N}_J := \{x \in \mathbb{Q}^n \mid |x_j - x_j^{\text{lp}}| \leq |x_j^{\text{inc}} - x_j^{\text{lp}}|, \, j \in J\}.$$

This neighborhood restricts lower and upper bounds of the general integer variables. Let $\mathcal{T} \subseteq \mathcal{S}_P$ denote a subset of currently available solutions, containing x^{inc}, to the MIP at hand. The DINS neighborhood can be written as a combination of a total of five neighborhoods

$$\begin{aligned}\mathcal{N}_{\text{DINS}} :=& \mathcal{N}_J \\ & \cap \mathcal{N}^{\text{fix}}\left(N_i \setminus J, x^{\text{inc}}\right) \\ & \cap \mathcal{N}^{\text{fix}}\left(N_b \cap \mathcal{M}^{=}(\{x^{\text{lp}}, x^{\text{root}}\} \cup \mathcal{T}), x^{\text{inc}}\right) \\ & \cap \mathcal{N}_{\text{LBranch}} \\ & \cap \mathcal{N}^{\text{obj}}(\delta, x^{\text{inc}}).\end{aligned}$$

The set of general integer variables outside of J is fixed to the values in the incumbent solution. Binary variables that have not changed between the root LP relaxation solution x^{root} and x^{lp} are fixed if they have taken the same value in all solutions in $\mathcal{T}$. Finally, the search is further restricted to a certain binary distance around the current incumbent solution through an additional local branching neighborhood. Among the possible choices for $\mathcal{T}$, the implementation of DINS for this work uses a set of up to five available solutions with best objective.

There is further work on LNS approaches that are not covered here. Note that the only heuristics that do not use an incumbent solution are RENS and Zero Objective. In [16], an extension of Local Branching has been proposed that starts from an infeasible reference point. Such points are quickly produced by rounding or with a few iterations of the Feasibility Pump [18]. In addition to the local branching constraint, the auxiliary problem of [16] is extended by additional variables to model and penalize the violation of constraints, inspired by the phase 1 of the Simplex algorithm. A recent approach called Alternating Criteria Search [31] also starts from infeasible reference points, and alternates between auxiliary problems with artificial feasibility objective and the original objective function of the input MIP in a parallel setting. The necessary diversification is obtained by fixing subsets of integer variables indexed by a random consecutive index set. Such a fixing scheme is a variant of Mutation [33] discussed in Section 2.1. The heuristics presented in [19] formulate and solve auxiliary problems only for the set N_i of general integer variables as a final post processing step after fixing all binary variables based on available, global problem structures such as cliques and implications between binary and integer variables.

3 Adaptive Large Neighborhood Search for MIP

The proposed Adaptive Large Neighborhood Search heuristic has as input the set of 8 available neighborhoods from Section 2 denoted by $\mathscr{H}$ (the set of actions). Table 1 gives a quick overview of the neighborhoods used, as well as their individual preconditions. In each call to the heuristic, ALNS basically performs the following steps.

1. Select a promising neighborhood $h_t \in \mathscr{H}$ and its associated auxiliary objective function $c_{\text{aux},t}$ via a *bandit selection strategy*.
2. Setup and solve the auxiliary problem given by $(h_t, c_{\text{aux},t})$.
3. Reward the neighborhood and update the bandit selection strategy for the next selection.

Different bandit selection strategies and their individual update procedures are subject of Section 4. The solution process of the auxiliary problem uses a strict limit on the number of branch-and-bound nodes to terminate the subproblem quickly and keep the overall computational effort small. It may still be very expensive to solve auxiliary problems if the neighborhood is large, especially since some neighborhoods do not fix integer variables directly. In Section 3.1, a generic approach is explained for fixing additional variables to reach any desired target fixing rate and hence reduce the subproblem complexity. Details on the dynamic adjustment of the target fixing rate and node limit are given in Section 3.2. Finally, Section 3.3 introduces the scoring mechanism to reward the selected neighborhood. Note that this work refers to the proposed scoring mechanism as "reward", but does not mention "penalties". Penalties are readily obtained by assigning small rewards.

3.1 Fixing and Unfixing Variables

Every neighborhood in Section 2 fixes a subset of integer variables to the values of a solution $x^{\text{sol}} \in \mathscr{S}_P$. The *fixed set* of a neighborhood $\mathscr{N}$ is defined as

$$\mathscr{M}^{\text{fix}} := \left\{ j \in \{1, \ldots, n_i\} \,|\, \mathscr{N} \subseteq \left\{ x \in \mathbb{Q}^n \,|\, x_j = x_j^{\text{sol}} \right\} \right\}.$$

The size of the fixed set is denoted by $n^{\text{fix}} = |\mathscr{M}^{\text{fix}}|$. Every neighborhood (action) $h \in \mathscr{H}$ operated by ALNS has a *target fixing rate* $\phi_{h,t} \in [0,1)$ that changes over time as will be explained in Section 3.2. It may happen that a selected neighborhood h_t does not reach its specified target fixing rate, ie.

$$n^{\text{fix}} < \phi_{h_t,t} \cdot n_i,$$

or that it fixes much more integer variables and hence restricts the search space more than necessary. ALNS treats both cases very similarly by using a generic variable fixing priorization to sort the set of possible (un)fixings.

Table 1: Overview of the neighborhoods that are used in ALNS. See Section 2 for information and references.

Neighborhood	Description	Preconditions
RINS	Fixes matching values in incumbent and LP relaxation solution	Feasible LP relaxation at current node, incumbent solution
Crossover	Fixes matching values in 2 or more randomly chosen, available solutions	Sufficiently many solutions
Mutation	Fixes random subset of variables to values in incumbent solution	Incumbent solution
RENS	Restricts the auxiliary problem to the feasible roundings around the current LP relaxation solution	Feasible LP relaxation at current node
Local Branching	Limits the maximal binary distance from the incumbent solution	Incumbent solution, MIP with binary variables
Proximity Search	Finds an improving solution that minimizes the binary distance from the incumbent	Incumbent solution, MIP with binary variables
Zero Objective	Reduces search for an (improving) solution to a feasibility problem	MIP with nonzero objective function
DINS	Sophisticated combination of RINS, Crossover, and Local Branching	Incumbent solution

In the first case, $\phi_{h_t,t} \cdot n_i - n^{\text{fix}}$ additional integer variables from $\overline{\mathscr{M}^{\text{fix}}} = \{1,\dots,n_i\} \setminus \mathscr{M}^{\text{fix}}$ have to be selected. For two variables $x_j, x_{j'} \in \overline{\mathscr{M}^{\text{fix}}}$ with reference solution values $x_j^{\text{sol}}, x_{j'}^{\text{sol}}$, the fixing $x_j = x_j^{\text{sol}}$ is preferred over $x_{j'} = x_{j'}^{\text{sol}}$, if, in decreasing order of priority,

1. x_j has a smaller distance than $x_{j'}$ from $\mathscr{M}^{\text{fix}}$ in the *variable constraint graph* (see below).
2. The reduced costs for fixing $x_j = x_j^{\text{sol}}$,

$$c_j^{\text{red}} \cdot \left(x_j^{\text{sol}} - x_j^{\text{root}_j}\right) < c_{j'}^{\text{red}} \cdot \left(x_{j'}^{\text{sol}} - x_{j'}^{\text{root}_{j'}}\right)$$

   are smaller than those for fixing $x_{j'} = x_{j'}^{\text{sol}}$.

3. The *pseudo costs* (see below) for fixing $x_j = x^{\text{sol}}$ are smaller than for $x_{j'} = x_{j'}^{\text{sol}}$,

$$\Psi_j\left(x_j^{\text{sol}} - x_j^{\text{root}}\right) < \Psi_{j'}\left(x_{j'}^{\text{sol}} - x_{j'}^{\text{root}}\right).$$

4. Randomly.

The rationale behind this variable priorization is to first keep the auxiliary problem connected by considering the *variable constraint graph*. For a given MIP P, the variable constraint graph G_P has the variables and constraints $\{v_j \,|\, j \in \{1,\dots,n\}\} \cup \{w_j \,|\, j \in \{1,\dots,m\}\}$ as nodes. The edges $E(G_P)$ correspond to the nonzero entries of the matrix A:

$$E(G_P) = \{(v_j, w_i) \,|\, a_{ij} \neq 0\}.$$

Distances in G_P are breadth first distances. If the original problem has a block structure, the variable priorization concentrates additional fixings on those blocks with a nonempty intersection in $\mathscr{M}^{\text{fix}}$. Related ideas are used, e.g., in presolving for detecting independent subproblems [20], or within the *Graph-Induced Neighborhood Search* (GINS) released with SCIP 4.0 [30].

The cost based scores in steps 2 and 3 both penalize a deviation of the potential fixing from an LP solution at the root node of the branch-and-bound search. The first associated penalty uses reduced costs. Reduced costs are recorded after every solved LP at the root node of the solving process. If they are higher than any previously recorded reduced costs for x_j, they are stored together with the corresponding LP solution value $x_j^{\text{root}_j}$. Therefore, the LP solution values used to compare the potential fixings of x_j and $x_{j'}$ may come from different LP solutions.

The pseudo-cost score treats the fixing as a branching and estimates the potential gain in the dual bound. Both variables are compared on the same, namely the final root LP solution on which the solution process started branching. *Pseudo costs* [4] are a common aggregate of branching information on the variables. Given the average increase Ψ_j^+ (Ψ_j^-) per unit fractionality of the dual bound after branching upwards (downwards) on a variable x_j for $j \in \{1,\dots,n_i\}$, the pseudo costs of j are a function

$$\begin{aligned} \Psi_j : \quad \mathbb{Q} &\to \mathbb{Q}_{\geq 0} \\ z &\mapsto \begin{cases} \Psi_j^+ \cdot z, & \text{if } z \geq 0 \\ -\Psi_j^- \cdot z, & \text{if } z < 0. \end{cases} \end{aligned} \tag{2}$$

As an example, assume that a binary variable x_j has an LP solution value $x_j^{\text{lp}} = 0.4$ and an incumbent solution value of 1. Assume that the average dual bound increase has been $\Psi_j^- = 10$ for branching down on x_j and $\Psi_j^+ = 5$ for branching up. The pseudo costs for the fixing $x_j = x_j^{\text{sol}}$ can be calculated as $\Psi_j(x_j^{\text{sol}} - x_j^{\text{lp}}) = \Psi_j^+ \cdot 0.6 = 3$. If the solution value had been 0 instead of one, the corresponding pseudo cost score is $\Psi_j(0 - x_j^{\text{lp}}) = -\Psi^- \cdot (-0.4) = 4$ for branching down on x_j.

Only slight details are changed if the neighborhood was too restrictive, such that $n^{\text{fix}} - \phi_{h_t,t} \cdot n_i$ variables from $\mathscr{M}^{\text{fix}}$ should be selected and unfixed (relaxed). Distances are now computed in the variable constraint graph starting from $\overline{\mathscr{M}^{\text{fix}}}$, and variables with a small distance are preferably relaxed to keep the auxiliary problem connected. Similarly to before, variables are relaxed preferably if they have a large reduced cost score or, in case a tie occurs, a large pseudo cost score. Finally, if none of the scores discriminate between two variables, the preference is given by a random score assigned to each variable. Generic (un-)fixing within ALNS is only applied if the target fixing rate $\phi_{h_t,t}$ is missed by a tolerance of 10 %, i.e. only if $n^{\text{fix}} \notin [(\phi_{h_t,t} - 0.1) \cdot n_i, (\phi_{h_t,t} + 0.1) \cdot n_i]$.

3.2 Dynamic Limits

Good limits on the computational budget of an LNS heuristic are essential to make it useful inside a MIP solver. To this end, a tradeoff must be made between the intensity of the search inside the auxiliary problem and the runtime. For this work, the complexity of the subproblem and the budget are adapted dynamically between the individual calls to ALNS.

All of the following dynamic decisions consider the auxiliary problem $P^{\text{aux}}(t)$ at the t-th round of ALNS ($t = 0, 1, \dots$) and its *solution status* $s\,(P^{\text{aux}}(t))$ which can be one of

- *inf*, if $P^{\text{aux}}(t)$ was infeasible,
- *opt*, if $P^{\text{aux}}(t)$ was solved to optimality
- *sol*, if $P^{\text{aux}}(t)$ provided an improving solution for P
- *nosol*, if no improving solution was found searching $P^{\text{aux}}(t)$.

Target fixing rate $\phi_{h,t}$ The first dynamic adjustment of the auxiliary problem complexity over time is described in [33], together with the introduction of the Crossover and Mutation LNS heuristics (cf. Section 2), which have been implemented as neighborhoods available to ALNS. The authors of [33] control the complexity of an auxiliary problem $P^{\text{aux}}(t)$ by specifying the amount of integer variables that should be fixed by a neighborhood. The intuition is that the complexity of $P^{\text{aux}}(t)$ decreases with increasing fixing rate. In the notation of the present work, the amount of fixed integer variables is specified by means of the target fixing rate $\phi_{h,t} \in [0, 1)$ of a neighborhood $h \in \mathscr{H}$ during round t of ALNS.

For $h \in \mathscr{H}$, let $T_h(t)$ denote the number of times that h has been selected, including round t. The fixing rate is modified according to the status in round t as

$$\phi_{h,t+1} = \begin{cases} \phi_{h,t}, & \text{if } h \neq h_t \text{ or } s\,(P^{\text{aux}}(t)) = \textit{sol}\,, \\ \max\{0.1, \phi_{h,t} - 0.75^{T_h(t)} \cdot 0.2\}, & \text{if } s\,(P^{\text{aux}}(t)) \in \{\textit{inf}, \textit{opt}\} \\ \min\{0.9, \phi_{h,t} + 0.75^{T_h(t)} \cdot 0.2\} & \text{if } s\,(P^{\text{aux}}(t)) = \textit{nosol} \end{cases}$$

If $P^{\text{aux}}(t)$ was too easy for the solver, ie. it could be solved to optimality or infeasibility within a given node budget, the fixing rate for the next iteration is decreased. If no new solution was found, the target fixing rate is increased. If a solution was found within the node limit, but the search could not be completed, the fixing rate is kept. The additive change of the fixing rate is 0.2 initially, which is multiplied with 0.75 after every update step, exactly as in [33]. The use of max and min ensures that the target fixing rate stays within 10 % and 90 %. In our implementation, those two values are parametrized and can be individually set for every neighborhood. Every neighborhood h is initialized with a conservative target fixing rate of $\phi_{h,1} = 0.9$ (respectively the user-defined maximum value), see Section 5 for details.

Node limit ν_t^{lim} The main budget limitation of ALNS is a limit on the number of branch-and-bound nodes. The node limit ν_{t+1}^{lim} for the next round of ALNS is adjusted based on the results of the auxiliary problem of round t as follows.

$$\nu_{t+1}^{\text{lim}} = \begin{cases} \nu_t^{\text{lim}}, & \text{if } s\left(P^{\text{aux}}(t)\right) \in \{opt, inf, sol\} \\ \min\{\lfloor \nu_t^{\text{lim}} \cdot 1.05 \rfloor + 1, 5000\}, & \text{if } s\left(P^{\text{aux}}(t)\right) = nosol \end{cases}$$

Similarly to other LNS heuristics in SCIP, ALNS uses an affine linear function of the branch-and-bound nodes ν^{bb} in the main search to limit the search effort inside auxiliary problems. Concretely, the next round t of ALNS is called as soon as the main search has made sufficient search progress such that

$$\kappa_0 + \frac{u(t-1)+1}{(t-1)+1} \cdot \kappa_1 \cdot \nu^{\text{bb}} - \sum_{i=1}^{t-1}(100 + \nu_i) \geq \nu_t^{\text{lim}}. \tag{3}$$

Here, κ_0 and κ_1 are positive coefficients, and $u(t-1)$ denotes the total number of improving solutions found by the end of round $t-1$. In 3, the linear growth of the budget is increased or decreased based on the total number of improving solutions that ALNS provided. With this strategy, ALNS slowly fades out if it does not find improving solutions. The last term expresses the total node budget used so far with an additional 100 nodes to account for the setup costs of $P^{\text{aux}}(i)$.

3.3 A Reward Function for Neighborhoods

All of the bandit selection strategies presented in Section 4 require the definition of a suitable reward function. Intuitively, the reward should always be higher for neighborhoods that improve over the current incumbent solution, but also depend on the achieved objective quality. Furthermore, a neighborhood that failed fast should be rewarded higher than an unsuccessful neighborhood whose execution required more of the budget. In order for some of the selection strategies from Section 4 to work correctly, we require that a reward should be in the interval $[0, 1]$. A reward of 0 is the worst possible score, i.e., the maximum penalty.

Let $h_t \in \mathscr{H}$ denote the selected neighborhood in round $t > 0$, and let $c^{\text{old}} := c^T x^{\text{inc}}$ denote the objective before the execution of h_t, if an incumbent solution $x^{\text{inc}} \in \mathscr{S}_P$ is available, or $c^{\text{old}} := \infty$, otherwise. Similarly, c^{new} is the objective of the best known solution after running h_t. Further, let ν_t^{lim} and ν_t denote the node limit and amount of nodes used by h_t, respectively, and let $n^{\text{fix}}(t)$ denote the number of integer variables fixed by h_t.

Two reward functions are combined to reward both the presence of a new incumbent solution and the objective improvement. The former is expressed by the *solution reward*

$$r^{\mathrm{sol}}(h_t,t) := \begin{cases} 1, & \text{if } s\left(P^{\mathrm{aux}}(t)\right) \in \{opt, sol\}, \\ 0, & \text{else.} \end{cases}$$

The improvement in solution quality is measured by the *closed gap reward*

$$r^{\mathrm{gap}}(h_t,t) := \frac{c^{\mathrm{old}} - c^{\mathrm{new}}}{c^{\mathrm{old}} - c^{\mathrm{dual}}},$$

which evaluates to 0 if no improving solution could be found, and is 1 if the new solution has an objective that is equal to the dual bound, i.e. that is optimal for P. As a convention, the closed gap reward is 1 if the neighborhood could contribute the first solution to the problem. Since most neighborhoods require a known solution as input (cf. 1), this is only possible with RENS and Zero Objective.

Since the time measurement in SCIP is not deterministic, we approximate the *effort* $\xi(t)$ spent on the search of the auxiliary problem by using the number of nodes as

$$\xi(t) = \left(1 - \frac{n^{\mathrm{fix}}(t)}{n_i}\right) \frac{\nu_t}{\nu_t^{\mathrm{lim}}}.$$

In order to compensate for different target fixing rates between different heuristics, $\xi(t)$ uses a scaling by the remaining number of free integer variables. The effort lies in the interval $[0,1]$. It is $\xi(t) = 1$ if and only if the node limit was exhausted ($\nu_t = \nu_t^{\mathrm{lim}}$) and no integer variables were fixed by the neighborhood. We propose to use the effort in two ways. If a neighborhood fails to produce a better solution, the last of the three individual reward functions is the *failure reward*

$$r^{\mathrm{fail}}(h_t,t) := \begin{cases} 1, & \text{if } s\left(P^{\mathrm{aux}}(t)\right) \in \{opt, sol\}, \\ 1 - \xi(t), & \text{else,} \end{cases}$$

which uses the effort directly and becomes smaller depending on the effort spent in an auxiliary problem, if no improving solution was found. With two additional convex combination parameters $\eta_1, \eta_2 \in [0,1]$, the reward function of ALNS combines all three rewards as

$$r^{\mathrm{alns}}(h_t,t) := \eta_1 r^{\mathrm{fail}}(h_t,t) + (1-\eta_1) \cdot \frac{\eta_2 r^{\mathrm{sol}}(h_t,t) + (1-\eta_2) r^{\mathrm{gap}}(h_t,t)}{1 + \xi(t)} \tag{4}$$

The first control parameter η_1 separates the reward between runs that were successful and runs that failed to improve the incumbent solution. The second parameter η_2 adjusts between the solution and the closed gap rewards. The result, which is again a reward in the interval $[0,1]$, is scaled by the effort involved to reward fast neighborhoods higher. Figure 1 depicts the individual elements of the ALNS reward function visually. For the computational experiments, $\eta_1 = 0.5$ and $\eta_2 = 0.8$ were used.

4 Selection Strategies for Multi Armed Bandit Problems

The goal of the present work is a framework that selects among the LNS heuristics presented in Section 2 and that tries to maximize their utility under a shared computing

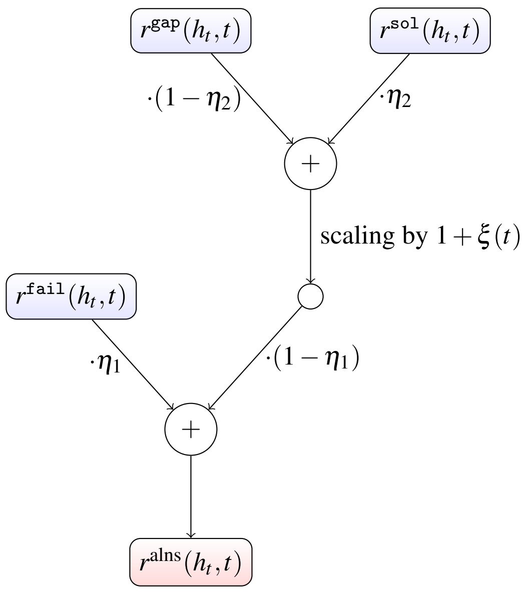

Figure 1: Diagram of the proposed reward function.

budget. Such a sequential decision process from a finite set of actions (heuristics) with unknown outcome appears in the literature as *Multi Armed Bandit Problem* [9].

The basic multi armed bandit problem can be described as a game, which is played over multiple rounds. In every round $t = 1, 2, \dots,$ the player chooses one action $h_t \in \mathscr{H}$ from a finite set of available actions. In return for playing h_t, the player observes a *reward* $r(h_t, t) \in [0, 1]$ for the selected action. The aim for the player is to maximize their total revenue $\sum_t r(h_t, t)$. Since only the reward of the selected action can be observed at a time, every suitable algorithmic strategy must find a good balance between exploration across all actions and exploitation of the best action seen so far.

Depending on the nature of the reward distribution, we distinguish between two main scenarios of multi armed bandit problems. In the *stochastic scenario*, the observable rewards $r(h, t)$ for every action $h \in \mathscr{H}$ are independent, identically distributed (i.i.d.) random draws over time from a probability distribution with unknown *expected reward* $\mu_h \in [0, 1]$. In the stochastic scenario, a good strategy should play an action h^* with maximum expected reward $\mu_{h^*} \geq \mu_{h'}$ $\forall h' \in \mathscr{H}$ as often as possible.

In the *adversarial scenario*, the player faces an opponent that chooses the rewards with the goal to maximize the player's *regret*–the discrepancy between the player's revenue and the best possible revenue. The opponent may take into account all choices previously made by the player, but does not know the selected action at time t. After the player and the opponent have each made their decisions, the player receives the reward $r(h_t, t)$ for the selected action only, while the opponent is informed about the player's choice h_t. It is noteworthy that in the adversarial scenario, the opponent has an incentive to play rewards different from 0 in every round of the game because the player's regret is minimal in every round t where all actions have a reward of 0. In the adversarial scenario, a good strategy must be necessarily randomized in some way because every deterministic algorithm is easily fooled by the opponent, who can minimize the player's total revenue by assigning a reward of 0 to the player's deterministic next action, and 1 to all other actions.

Intuitively, the adversarial scenario seems much harder to approach than the stochastic scenario because the latter is indifferent to choices made by the player, and estimates of the expected rewards can be built over time. It turns out that it is possible, even for the adversarial scenario, to create strategies that yield an asymptotically optimal revenue in their respective scenario. The reader is referred to the survey [9] for more information about and variants of the following algorithms.

4.1 Strategies for the Stochastic Scenario

The assumption of the stochastic scenario is that the observable reward of each action $h \in \mathscr{H}$ is the realization of a random variable R_h around its expected value

$\mu_h = \mathbb{E}(R_h)$. Let $T_h(t) := \sum_{i=1}^{t} \mathbb{1}_{h_i=h}$ denote the number of times that action h has been selected until round t. The *sample average*

$$\bar{r}_h(t) := \frac{1}{T_h(t)} \sum_{i=1}^{t} \mathbb{1}_{h_i=h} r(h,i)$$

is an unbiased estimator of the unknown quantity μ_h.

Algorithm 1: ε-greedy

```
   Input: Set of actions $\mathscr{H}$, parameter $\varepsilon \geq 0$
1  $t \leftarrow 0$
2  while not stopped do
3      $t \leftarrow t+1$
4      $\varepsilon_t \leftarrow \varepsilon \cdot \sqrt{\frac{|\mathscr{H}|}{t}}$
5      Draw $v_t \sim \mathbb{U}([0,1])$
6      if $v_t \leq \varepsilon_t$ /* Selection of next action          */
7       then
8          Draw $h_t \sim \mathbb{U}(\mathscr{H})$
9      else
10         $h_t \leftarrow \operatorname{argmax}_{h \in \mathscr{H}} \bar{r}_h(t)$
11     Update $\bar{r}_{h_t}(t+1)$ by the observed reward $r(h_t,t)$
```

Algorithm 1 is a very simple, randomized selection strategy for the multi armed bandit problem. It uses the short notation $\mathbb{U}(X)$ to denote the *uniform distribution* over a set X. The initial lack of reward information is compensated by a randomized selection of the first few actions. The amount of random selections decreases at the speed of $\frac{1}{\sqrt{t}}$ and can be controled by the input parameter ε. With increasing t, it therefore becomes less and less likely to choose an action at random, whereas the probability of greedily exploiting the best action increases.

A different, deterministic approach to the stochastic scenario is based on the principle of optimism at the face of uncertainty. The approach is called *Upper Confidence Bound (UCB)* algorithm. Under the assumption that all rewards are in the interval $[0,1]$, it holds with a probability of at least $1-\delta$ that

$$\bar{r}_h(t) + \sqrt{\frac{2}{T_h(t)} \ln \frac{1}{\delta}} > \mu_h. \tag{5}$$

The quantity on the left side of Equation 5 is an upper confidence bound for μ_h at confidence level δ. With the goal to ultimately find the action h^* with maximum expected reward μ_{h^*}, the UCB algorithm selects the action that maximizes the upper confidence bound instead, while increasing the confidence level as a function of time, e.g., $1/\delta = 1+t$. The rationale behind this is that also inferior actions become more attractive to the algorithm after they have not been selected for a while. The width of the confidence interval is further controlled by a parameter $\alpha \geq 0$. Let H be an ordered tuple of the elements in $\mathscr{H}$. The selection strategy α-UCB selects

$$h_t \in \begin{cases} \underset{h \in \mathscr{H}}{\operatorname{argmax}} \left\{ \bar{r}_h(t-1) + \sqrt{\frac{\alpha \ln(1+t)}{T_h(t-1)}} \right\} & \text{if } t > |\mathscr{H}|, \\ \{H_t\} & \text{if } t \leq |\mathscr{H}|. \end{cases} \tag{6}$$

The case distinction in Equation 6 is necessary to obtain a meaningful initialization of all sample means and because $T_h(|\mathscr{H}|+1) \geq 1$ for all $h \in \mathscr{H}$ is required for Equations 5 and 6 to be well defined. In the first case of Equation 6, eventual occuring ties are broken uniformly at random. The special case of $\alpha = 0$ yields a completely greedy exploration strategy that does not take into account the upper confidence bound.

4.2 A Strategy for the Adversarial Scenario

Algorithm 2 is the standard algorithmic strategy for adversarial multi armed bandit scenarios. It is briefly called Exp.3, which is an abbreviation of "Exponential Weight Algorithm for Exploration and Exploitation". In each round t, the next action is selected randomly from a probability distribution defined by marginal probabilities $p_{h,t}$ for each $h \in \mathscr{H}$. After receiving the reward $r(h_t,t)$, the weight update is performed in two steps. First, the cumulative reward of the selected action h_t is updated in line 7. The cumulative reward weighs the observed reward with the probability to select h_t, thereby emphasizing actions with a high reward compared to their current selection probability. Second, the probabilities for the next iteration $t+1$ are computed as a convex combination of two probability distributions based on the choice of γ. In the two extreme cases $\gamma \in \{0,1\}$, the algorithm either draws from a uniform distribution ($\gamma = 1$) in the next iteration, or from a distribution defined by the normalized exponential function of the cumulative rewards ($\gamma = 0$).

Algorithm 2: Exp.3

```
  Input: Set of actions $\mathscr{H}$, convex combination parameter $\gamma \in [0,1]$
1 $p_{h,1} \leftarrow \frac{1}{|\mathscr{H}|}$, $Q_h \leftarrow 0 \ \forall h \in \mathscr{H}$
2 $t \leftarrow 0$
3 while not stopped do
4     $t \leftarrow t+1$
5     Draw $h_t$ from probability distribution $p_{h,t}$
6     Observe reward $r(h_t,t)$
7     $Q_{h_t} \leftarrow Q_{h_t} + \frac{r(h_t,t)}{p_{h,t}}$
8     foreach $h \in \mathscr{H}$ do
9         $p_{h,t+1} \leftarrow (1-\gamma)\frac{\exp(Q_h)}{\sum_{h'}\exp(Q_{h'})} + \frac{\gamma}{|\mathscr{H}|}$
```

Table 2: Overview of involved parameters and values for the simulation and the MIP experiments. All SCIP parameters are preceded by `heuristics/alns/`. The placeholder `*` must be substituted by the name of a neighborhood, e.g., `rens`.

Symbol	SCIP parameter(s)	Section Ref.	Simulation	MIP
δ	`minimprov{low,high}`	2	0.01	0.01
k	`crossover/nsols`	2.1	2	2
$d_{\max}$	not parameterized	2.2	$0.2 \cdot n_b$	$0.2n_b$
$\lvert\mathscr{T}\rvert$	`dins/npoolsols`	2.2	5	5
v_1^{lim}	`alns/minnodes`	3.2	50	50
κ_0	`alns/nodesofs`	3.2	500	500
κ_1	`alns/nodesquot`	3.2	0.1	0.1
η_1	`rewardbaseline`	3.3	0.5	0.5
η_2	`rewardcontrol`	3.3	0.8	0.8
$\phi_{h_t,t}$	`*/{min,max}fixingrate`	3.1	$\{0.1, 0.3, \ldots, 0.9\}$	dynamic in $[0.3, 0.9]$
ε	`epsilon`	4.1	–	0.4685844
α	`alpha`	4.1	–	0.0016
γ	`gamma`	4.2	–	0.07041455
–	`banditalgo`	4	–	{Exp.3,UCB,ε-greedy}

5 Computational Results

The proposed ALNS framework has been implemented and tested as an additional plugin on top of SCIP 5.0, using CPLEX 12.7.1 as the underlying LP solver. All 8 neighborhoods listed in Table 1 have been incorporated into ALNS. As instance set, we use the union of three MIPLIB collections 3.0, 2003, and 2010,[7, 3, 27] and the Coral [11] instance set, totaling to 666 instances. The computational experiments for the present work are split into two parts. The first part is an offline simulation that uses reward information about all neighborhoods in each call to ALNS. This information is used to compare neighborhoods directly, and to calibrate the parameters of the bandit selection strategies from Section 4. Section 5.3 describes the results that we obtained with the ALNS framework inside of SCIP using the readily calibrated selection strategies. Since a lot of parameters have been introduced in the previous sections, Table 2 summarizes the parameter settings used for the simulation and the performance experiments in this section.

5.1 Neighborhood Comparison

The first part aims at providing a fair comparison between the neighborhoods that have been implemented in the ALNS framework. Instead of choosing a single neighborhood at each call, all neighborhoods are executed one after another, and their individual rewards are recorded. In order to ensure fairness, every found improving solution is only recorded for the reward function. SCIP does not receive a solution as this would impact consecutive neighborhoods. All dynamic decisions are deactivated for this experiment. The target fixing rate is kept fixed at $0.1 - 0.9$ in steps of 0.2, with a tolerance of ± 0.1. Recall that the additional generic fixings/unfixings are only applied if the obtained fixing rate lies outside of the tolerance interval. The experiments have been conducted on a Linux cluster using Ubuntu 16.04, with a time limit of 5h for each instance.

Not all neighborhoods can be used for all problems. Local Branching and Proximity can only be executed for instances involving binary variables. Zero Objective requires a nonzero objective function. Furthermore, DINS requires a solution pool

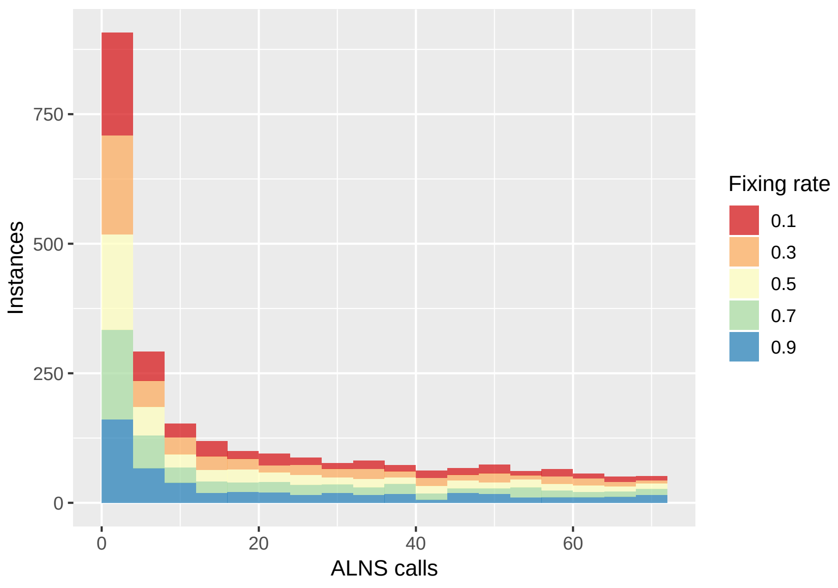

Figure 2: Histogram of call frequencies.

Table 3: Number of successful calls of ALNS

Fixing rate	Calls	Success	Succ. Rate
0.1	9037	841	0.093
0.3	9233	975	0.106
0.5	9789	1005	0.103
0.7	9925	1196	0.121
0.9	10085	1337	0.133

with at least 5 solutions, while RENS and RINS require a feasible LP relaxation at the local node. For this experiment, the ALNS framework is only executed if all those conditions are met.

Among the total set of instances, 494 instances allowed to execute all 8 neighborhoods at least once. A histogram of the number of calls is shown in Figure 2 with a bin width of 4 calls. One observes that for more than half of the instances, the heuristic is called at most eight times. This means that a UCT selection algorithm would have tried every neighborhood at most once in the real setup. On the other hand, the initialization phase has been finished on more than 200 instances. On average, ALNS was executed between 18.3 and 20.4 times per instance, depending on the fixing rate.

As one expects, the running time spent in a sub-MIP decreases with the fixing rate. Therefore, the number of times that ALNS could be executed is different for every fixing rate. Table 3 shows the number of finished calls to the framework for every tested fixing rate. It ranges from 9037 at a fixing rate of 0.1 to 10085 at 0.9. Particularly interesting are the calls where at least one of the tested neigborhoods finds a solution. The number of successful calls of an oracle neighborhood is shown in total in column “Success” and as a rate in column “Succ. Rate”. The success rate of

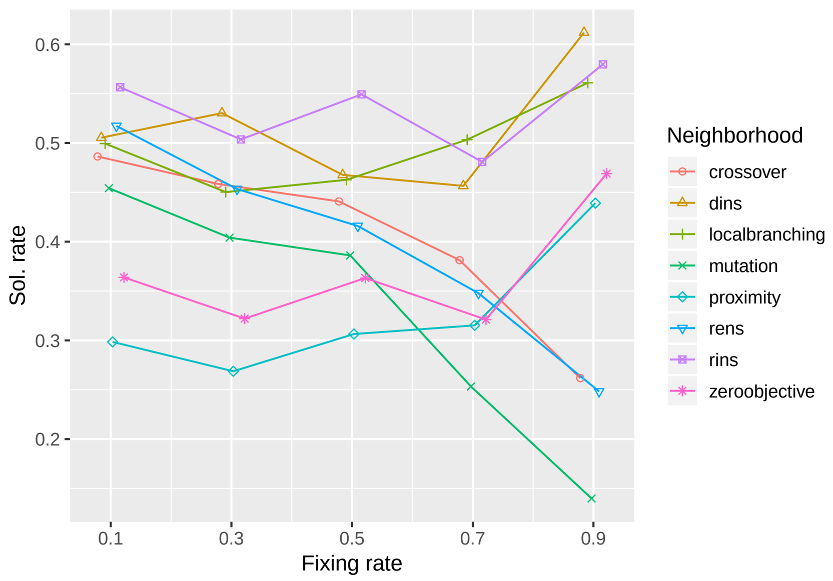

Figure 3: Solution rate by neighborhood and fixing rate.

the oracle neighborhood ranges from 9.3% to 13.3% and increases with an increasing fixing rate.

In all other calls, the selection process is only required to select a neighborhood that fails fast, but cannot contribute to the overall search process with an incumbent solution. Therefore, all results reported in the remainder of this section are reported for the subset of calls where at least one neighborhood finds a solution.

For this set of calls, Figure 3 shows the average number of solutions for every heuristic at every fixing rate. It can be observed that RINS, DINS, and Local Branching are almost consistently the top three neighborhoods across all tested fixing rates. While the solution frequency for those neighborhoods does not show monotonous trends, the solution rates for Crossover, RENS, and Mutation are decreasing with an increasing fixing rate.

The ranking between the neighborhoods is similar for the obtained rewards illustrated in Figure 4. Here, the average reward increases with the fixing rate. This is partly because the reward definition penalizes neighborhoods with a high fixing rate less strictly. A higher average reward results from an increased solution frequency, a better solution quality, and/or less effort to solve the subproblem. The neighborhood rewards at a particular fixing rate can be compared well.

As expected from the previous analysis, RINS, Local Branching, and DINS reach high average rewards. The highest increase in average reward can be observed for the Proximity and Zero Objective neighborhoods. At a high fixing rate of 0.9, RENS achieves the smallest average reward. The reward for Mutation only increases up to a fixing rate of 50%. Its reward is almost constant for all fixing rates $\geq$ 50%. RENS lacks a reference solution for additional, generic fixings, which is why it can run less frequently than others. A possible explanation for the decreasing scores of Crossover is the random selection of reference solutions. Searching a narrow neighborhood of a reference solution far away from the incumbent may lower its chances to find a better solution. The lower scores of Mutation are remarkable because RINS and Mutation

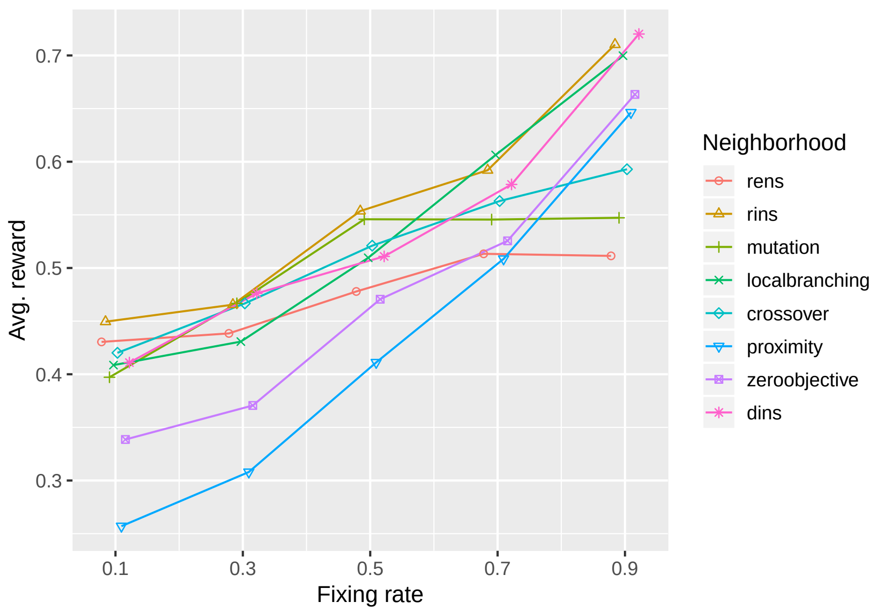

Figure 4: Avg. reward by neighborhood and fixing rate.

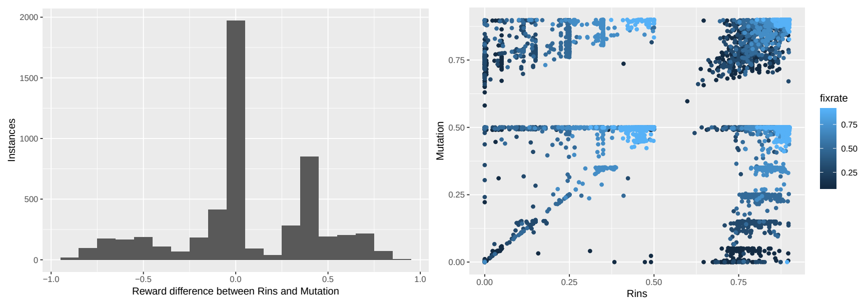

Figure 5: Reward comparison of RINS and Mutation.

use the same reference solution, namely the incumbent. RINS may even need additional generic fixings to reach higher target fixing rates, whereas the Mutation scheme always fixes the targeted percentage of integer variables. The large discrepancy in their solution rates indicates that more informed approaches such as the LP driven neighborhoods RINS or DINS are the most important fixing schemes.

One may ask the question whether a well-performing neighborhood such as RINS entirely dominates the less well-performing neighborhoods such as Mutation. Figure 5 illustrates the measured rewards for RINS and Mutation, respectively. RINS has a clear tendency to score higher, especially at larger fixing rates. However, also the execution of Mutation can be beneficial, as it reaches a higher reward in about 30% of the cases. Analogous comparisons for other pairs of neighborhoods yield similar results. Based on these observations, it is reasonable to enable all 8 neighborhoods by default, and to rely on the selection mechanism.

5.2 Simulation of the Selection Process

The rich data set from the previous section is now used for an offline calibration of the three bandit selection strategies. Each of the three bandit selection methods presented in Section 4 has a single parameter that can be calibrated for the use inside the ALNS selection process. Therefore, all three UCB, Exp.3, and ε-greedy are run on the entire collected data (all calls with and without at least one solution) with the aim to maximize the average reward obtained. Average rewards are computed over 100 repetitions of each algorithm. The simulation of the selection routines has been implemented in the programming language R.

Ideally, the selection performs better than a pure random selection for instances that allow for a certain number of calls to initialize the selection process. Note that certain parameter choices of the Exp.3 ($\gamma = 1$) and ε-greedy bandits are equivalent to a uniform random selection.

Figure 6 shows the selection quality for the three selection methods in terms of the average reward and solution rate. As in the previous section, the figures in this section refer to the subset of calls with at least one successful neighborhood. This means that the solution rate of an optimal selection algorithm would be a horizontal line with value 1.0 across all fixing rates. The first row depicts the selection quality of Exp.3 at different values of the γ parameter. Some hand-picked values $\{0.05, 0.5, 0.95\}$ are compared to $\gamma = 0.07041455$, the optimal value for γ computed by the R function `optimize`, and a pure random selection named "avg". At all tested values of the γ-parameter, the selection quality of Exp.3 is better than purely random. Furthermore, the experiment reveals that higher values of γ decrease the selection quality across all tested fixing rates. The choice of $\gamma = 0.95$ shows, as expected, almost the same selection quality as a pure random selection.

The best choice for ε computed by R is 0.4685844. The results for $\varepsilon = 4$ are indistinguishable from random sampling. The reason is the limited number of calls to ALNS, for which $\varepsilon_t \geq 1$ for all $t \leq 71$, such that only random sampling is applied by the ε-greedy algorithm.

Finally, the average reward of the UCB selection algorithm has been maximized for the parameter choice of $\alpha = 0.0016$. Its selection quality is depicted in the last row of Figure 6. It can be noted that the average selection quality is higher for UCB and ε-greedy than for Exp.3. Especially the solution rate is maximized by the UCB algorithm. The optimal values for the different parameters can be interpreted as follows. The optimal value for the γ-parameter is very close to a purely weight based Exp.3 selection strategy. The optimal value of the α-parameter shows a higher selection quality than the nearby value of $\alpha = 0$, a purely greedy selection. This is seconded by the optimal value of the ε-parameter. This shows that learning from past observations clearly helps the selection process at later stages.

Another observation is that the plots of Figure 6 seldomly cross, i.e. the ranking between different parameter choices is the same for different fixing rates. This indicates that the selection algorithm can be safely combined with an adaptive fixing rate.

The learning success of the bandit selection methods is depicted in Figure 7. It shows the average solution rate of Exp.3 as a function of the number of calls to the ALNS framework for all four different choices of the parameter γ. As a comparison serves the average solution rate per call. The picture illustrates that the solution rate for the choice of $\gamma = 0.07041455$ is better than random after a small number of calls. Furthermore, while the average solution rate stays relatively constant around 0.3, the solution rate of Exp.3 has a clear tendency to increase with the number of calls. It

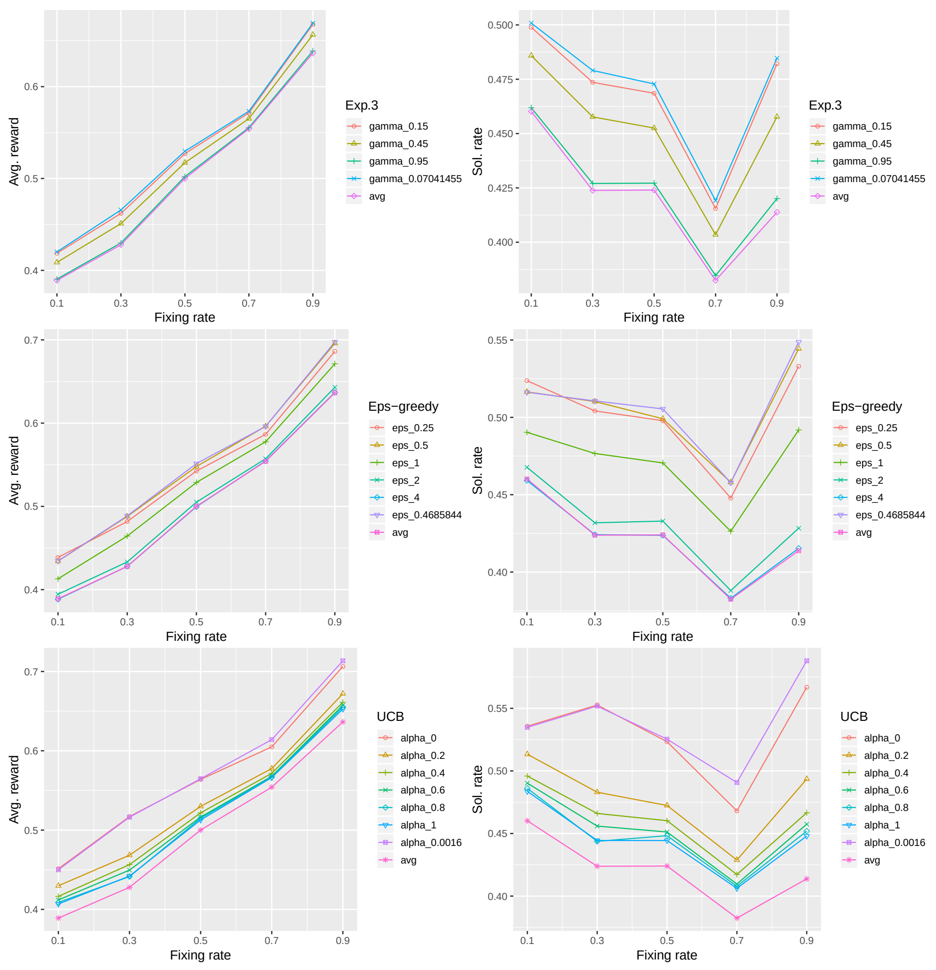

Figure 6: Comparison of selection performance for different parameter choices.

reaches a solution rate of 0.6 after 63 calls.

The two other bandit algorithms achieve this mark much earlier. The second diagram shows the solution rate by call for the ε-greedy algorithm for two choices of ε. Compared to Exp.3, the selection achieves an even better solution rate. Its margin from the average solution rate is higher. As an example, the (arbitrary) mark of a solution rate of 0.6 is first reached after 24 calls, and reliably surpassed after 40 calls to the selection routine.

The final plot shows that UCB outperforms the two other bandit selection routines. The mark of 0.6 is reached after 17 calls for the first time, and almost consistently surpassed after 30 calls. The price for this selection performance is that the first 8 observations must be spread over the 8 neighborhoods to select from, which is why UCB achieves exactly average performance at this early stage. At a later stage, UCB reaches a solution rate of 1.0 for the four rightmost observations, i.e., UCB can safely identify and select a well performing neighborhood at this stage. Recall that these plots represent average solution rates over 100 repetitions of the experiment.

As a conclusion, all three bandit selection algorithms achieve an above average selection performance, as desired. With an increasing initialization time, the learning effect becomes more pronounced. UCB achieves the best solution rate, followed by ε-greedy and Exp.3. Arguably, the good solution rate is an indication that the designed reward function, which the bandits actually receive, captures the ranking between the neighborhoods sufficiently well within the ALNS framework.

5.3 MIP Performance

This section presents the computational benefit of ALNS inside SCIP in a real setting where only one neighborhood can be called at a time. All 3 bandit selection methods have been tested and compared to SCIP with deactivated ALNS. The individual selection parameters are set to their individual optimal value as described in the previous section. The individual, existing Large Neighborhood Search heuristics RENS, RINS, and Crossover are active independently from ALNS in all four settings, as this represents the default settings of SCIP 5.0 with CPLEX 12.7.1 as LP solver. This experiment has been conducted on a Linux cluster of 32 computing nodes equipped with Intel Xeon CPU E5-2670 v2 at 2.50GHz. The time limit was 2h for every instance. In order to measure time as accurately as possible, every job has been scheduled exclusively.

Table 4 shows aggregated results for two performance measures, the solving time to optimality and the primal integral. It has been prepared using the Interactive Performance Evaluation Tools [23]. Individual outcomes for every instance and setting are found in Table 5 in the appendix. The measures are presented as shifted geometric mean time (using a shift of 1 sec.) and average primal integral. For a better quantitative assessment, relative percentage deviations from the fastest setting are shown. Table 4 summarizes the performance for the entire test bed as well as three interesting subgroups. The first subgroup "Diff" contains all instances on which activating ALNS has an effect on the solving process of SCIP for at least one selection strategy. On the complementary group "Equal", ALNS does not alter the solving process. A change in the path is detected by a change in the number of LP iterations of the main solution process. Both groups drop all instances that could be solved by none of the four settings. The last subgroup is the subset of MIPLIB 2010 [27] benchmark instances.

The overall fastest setting in this experiment is ALNS equipped with an Exp.3 selection algorithm. On the entire test bed, an approximate 2 % time improvement can be observed. Also with a UCB or ε-greedy selection, a time improvement can be

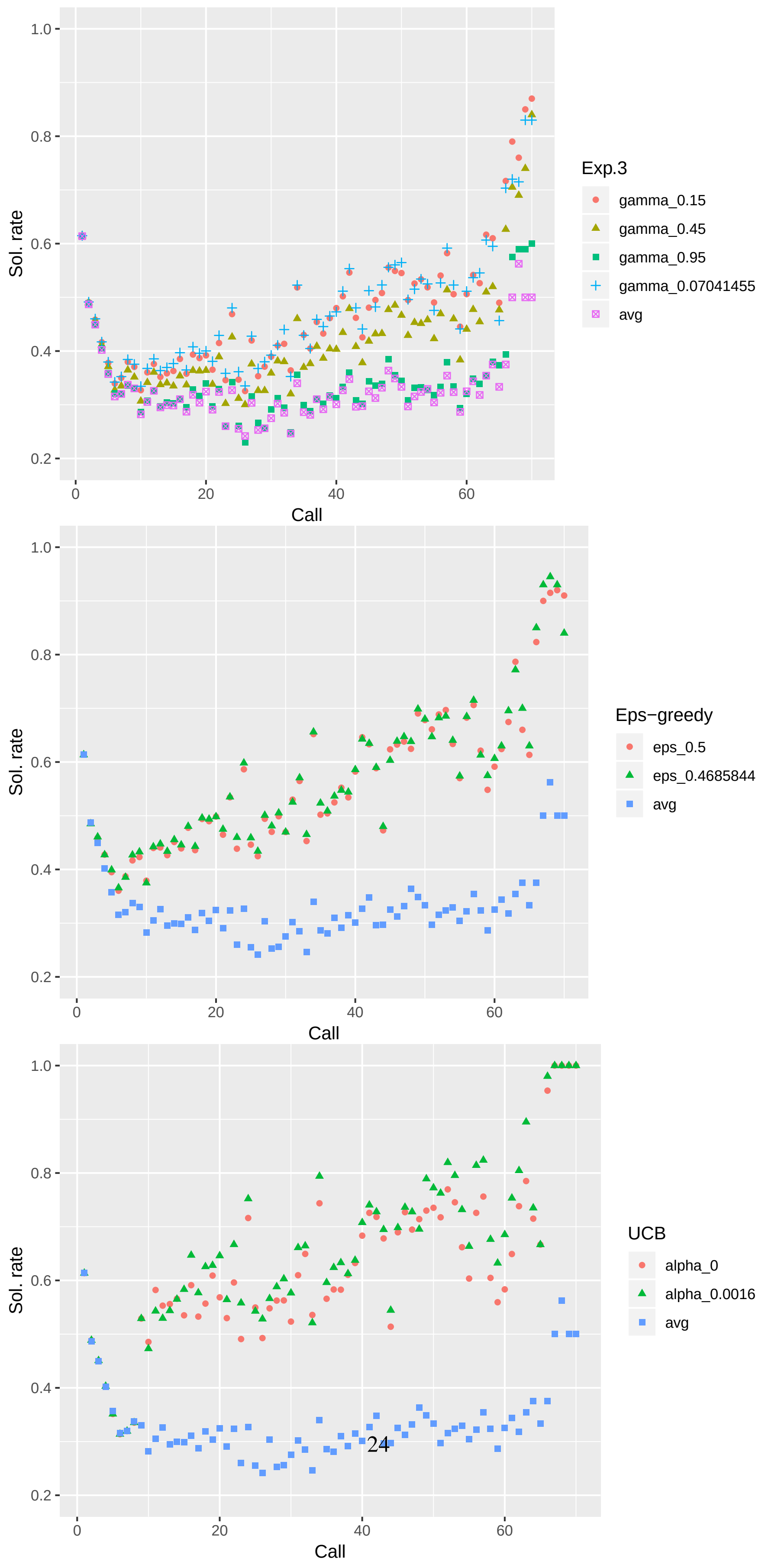

Figure 7: Average solution rate as a function of the individual call.

Table 4: Performance results of ALNS with different selection algorithms

Group	Setting	Inst.	Tlim	Time	Rel. Time %	Integral	Rel. Int. %
All	Exp.3	665	263	358.61	-1.89	77204.62	-2.11
All	Eps-greedy	665	266	360.90	-1.26	78001.49	-1.10
All	UCB	665	264	360.17	-1.46	76516.88	-2.98
All	ALNS off	665	265	365.52	0.00	78868.44	0.00
Diff	Exp.3	110	2	136.40	-11.39	10281.80	-34.92
Diff	Eps-greedy	110	5	137.21	-10.87	12278.62	-22.27
Diff	UCB	110	3	137.55	-10.65	10046.82	-36.40
Diff	ALNS off	110	4	153.94	0.00	15797.51	0.00
Equal	Exp.3	293	0	35.05	0.19	12888.25	0.23
Equal	Eps-greedy	293	0	35.49	1.46	12876.27	0.14
Equal	UCB	293	0	35.29	0.88	12870.50	0.10
Equal	ALNS off	293	0	34.98	0.00	12858.12	0.00
MIPLIB2010	Exp.3	87	6	314.01	-4.59	21610.16	-23.68
MIPLIB2010	Eps-greedy	87	7	321.02	-2.46	23195.39	-18.08
MIPLIB2010	UCB	87	6	323.58	-1.69	21097.42	-25.49
MIPLIB2010	ALNS off	87	7	329.12	0.00	28315.92	0.00

observed, albeit smaller. The time improvement is more pronounced for MIPLIB 2010, where ALNS with Exp.3 is more than 2 % faster than its bandit competitors and 4.6 % faster than SCIP without ALNS. Finally, for the set of affected instances, the ALNS time improvement (using Exp.3) is more than 11 %. A very similar time result has been obtained for the other two selection algorithms.

The primal integral is minimized by ALNS with the UCB selection strategy, closely followed by Exp.3. Overall, UCB achieves an improvement of 3 %, but the improvement is substantial for the subgroups MIPLIB2010 and Diff, where improvements of more than 25 % and 36 % are obtained, respectively. The ε-greedy selection also yields improved primal integrals for all sets of instances, especially on the set of affected instances. However, the improvements are less pronounced than for the other bandit algorithms. The subgroup “Equal” shows that the primal integral is almost not affected by ALNS if it does not affect the solution process. This indicates that the overhead caused by ALNS is negligible in case its solutions do not contribute to the overall search progress. Note that it is possible that the search is not affected even though ALNS finds a solution.

Overall, ALNS improves SCIP with all tested selection strategies with respect to both time and primal integral. The positive outcome for Exp.3 is unexpected because Exp.3 scored worse than UCB and ε-greedy in the simulation experiment. A possible explanation is the different scenario assumption of Exp.3. The important difference between the simulation and the MIP performance results is that solutions found by ALNS during the simulation experiment have been discarded and only reward information was kept. It is not unlikely that during the simulation, repeated calls to the same neighborhood resulted in the same solution. Designed for the stochastic scenario, UCB and ε-greedy aim at identifying a single, best performing neighborhood, which may give them an advantage over Exp.3 only in the simulation. The weighted randomized selection of Exp.3 may reduce the initial set of neighborhoods over time to two or three equally strong neighborhoods for an instance, or even continue using all available neighborhoods if they achieve similar rewards.

6 Conclusion

This article introduces Adaptive Large Neighborhood Search for MIP, a framework around eight well-known LNS neighborhoods from the literature. It has been implemented as a primal heuristic in SCIP and is publicly available since SCIP 5.0. The framework combines a selection procedure, which is governed by algorithms for the multi armed bandit problem, and the idea of generic additional variable fixings to adjust the complexity of the subproblems, as needed. We propose a reward function that combines the important aspects of solution quality and subproblem effort into a single number. We have used a simulation experiment to calibrate each bandit algorithm individually. Training the bandit algorithms with this reward function shows a clear trend to improve the solution rate with an increasing number of calls. As a byproduct of the simulation, we could see clear differences between the neighborhoods regarding the number of solutions they produce. Two of the neighborhoods that are most successful in our experiments, DINS and Local Branching, have been previously inactive in SCIP.

We currently see two future perspectives for this work. Adaptive algorithm selection may also be beneficial in other parts of the solver where the choice between similar methods largely affects the overall performance of the solver. In a recent article [25], promising results are presented for diving heuristics, and for dynamic switching between different pricing strategies of the Dual Simplex procedure to maximize the node throughput during the search. On the other hand, further algorithmic improvements regarding the use of auxiliary problems in general can be more easily incorporated into a single framework, such as the transfer of variable histories or conflict clauses, which may be very useful for the remaining search process.

Acknowledgements

The work for this article has been conducted within the Research Campus MODAL funded by the German Federal Ministry of Education and Research (BMBF grant number 05M14ZAM). The author would like to thank his colleagues Timo Berthold and Thomas Schlechte for their reviews and comments on earlier versions of this manuscript.

References

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

[2] T. Achterberg, T. Berthold, and G. Hendel. Rounding and propagation heuristics for mixed integer programming. In D. Klatte, H.-J. Lüthi, and K. Schmedders, editors, *Operations Research Proceedings 2011*, pages 71–76. Springer Berlin Heidelberg, 2012.

[3] T. Achterberg, T. Koch, and A. Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):1–12, 2006.

[4] M. Bénichou, J.-M. Gauthier, P. Girodet, G. Hentges, G. Ribière, and O. Vincent. Experiments in mixed-integer programming. *Mathematical Programming*, 1:76–94, 1971.

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

[6] T. Berthold. RENS–the optimal rounding. *Mathematical Programming Computation*, 6(1):33–54, 2014.

[7] R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, 58:12–15, 1998.

[8] R. Borndörfer, H. Hoppmann, and M. Karbstein. A configuration model for the line planning problem. In D. Frigioni and S. Stiller, editors, *ATMOS 2013 - 13th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, volume 33, pages 68 – 79, 2013.

[9] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *CoRR*, abs/1204.5721, 2012.

[10] COIN-OR branch-and-cut MIP solver, 2016. `https://projects.coin-or.org/Cbc`.

[11] Coral mip benchmark library, 2016. http://coral.ise.lehigh.edu/data-sets/mixed-integer-instances.

[12] IBM ILOG CPLEX Optimizer, 2016. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

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

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

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

[16] M. Fischetti and A. Lodi. Repairing milp infeasibility through local branching. 35:1436–1445, 05 2008.

[17] M. Fischetti and M. Monaci. Proximity search for 0-1 mixed-integer convex programming. *Journal of Heuristics*, 20(6):709–731, Dec 2014.

[18] M. Fischetti and D. Salvagnin. Feasibility pump 2.0. *Mathematical Programming Computation*, 1(2):201–222, Oct 2009.

[19] G. Gamrath, T. Berthold, S. Heinz, and M. Winkler. Structure-based primal heuristics for mixed integer programming. In *Optimization in the Real World*, volume 13, pages 37 – 53. 2015.

[20] G. Gamrath, T. Koch, A. Martin, M. Miltenberger, and D. Weninger. Progress in presolving for mixed integer programming. *Mathematical Programming Computation*, 7(4):367–398, 2015.

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

[22] A. Gleixner, M. Bastubbe, L. Eifler, T. Gally, G. Gamrath, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. E. Lübbecke, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, C. Schubert, F. Serrano, Y. Shinano, J. M. Viernickel, M. Walter, F. Wegscheider, J. T. Witt, and J. Witzig. The scip optimization suite 6.0. Technical Report 18-26, ZIB, Takustr. 7, 14195 Berlin, 2018.

[23] G. Hendel. IPET interactive performance evaluation tools. `https://github.com/GregorCH/ipet`.

[24] G. Hendel. New rounding and propagation heuristics for mixed integer programming. Bachelor thesis, 2011.

[25] G. Hendel, M. Miltenberger, and J. Witzig. Adaptive algorithmic behavior for solving mixed integer programs using bandit algorithms. Technical Report 18-36, ZIB, Takustr. 7, 14195 Berlin, 2018.

[26] E. B. Khalil, B. Dilkina, G. Nemhauser, S. Ahmed, and Y. Shao. Learning to run heuristics in tree search. In *26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[27] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011.

[28] P. Laborie and D. Godard. Self-adapting large neighborhood search: Application to single-mode scheduling problems. In P. Baptiste, G. Kendall, A. Munier, and F. Sourd, editors, *MISTA-07*, 08 2007.

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

[30] S. J. Maher, T. Fischer, T. Gally, G. Gamrath, A. Gleixner, R. L. Gottwald, G. Hendel, T. Koch, M. E. Lübbecke, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, S. Schenker, R. Schwarz, F. Serrano, Y. Shinano, D. Weninger, J. T. Witt, and J. Witzig. The scip optimization suite 4.0. Technical Report 17-12, ZIB, Takustr. 7, 14195 Berlin, 2017.

[31] L.-M. Munguía, S. Ahmed, D. A. Bader, G. L. Nemhauser, and Y. Shao. Alternating criteria search: a parallel large neighborhood search algorithm for mixed integer programs. *Computational Optimization and Applications*, 69(1):1–24, Jan 2018.

[32] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403 – 2435, 2007.

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

[34] A. Sabharwal, H. Samulowitz, and C. Reddy. Guiding combinatorial optimization with UCT. In N. Beldiceanu, N. Jussien, and E. Pinson, editors, *CPAIOR*, volume 7298 of *Lecture Notes in Computer Science*, pages 356–361. Springer, 2012.

[35] Xpress. FICO Xpress-Optimizer, 2016. `http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx`.

Appendix

Table 5: The MIP performance results from Section 5.3 for every instance. The 3 columns 2010, Diff, and Equal indicate the group membership of an instance to the respective group in Table 4.

ProblemName	2010	Diff	Equal	Time (sec.)				Prim. Int.			
				Exp.3	UCB	ε-greedy	ALNS off	Exp.3	UCB	ε-greedy	ALNS off
10teams			✓	13	13	13	12	558	588	599	606
22433			✓	3	3	3	3	290	270	270	270
23588			✓	6	6	6	6	212	202	222	202
30n20b8	✓		✓	256	254	255	254	15 902	15 736	15 769	15 814
50v-10				7 200	7 200	7 200	7 200	2 921	2 916	2 912	2 970
Test3			✓	5	5	5	5	195	195	195	195
a1c1s1				7 200	7 200	7 200	7 200	3 913	2 832	3 189	2 830
acc-tight4			✓	170	170	170	171	17 000	17 000	17 000	17 100
acc-tight5	✓		✓	118	116	117	116	11 800	11 600	11 700	11 600
acc-tight6			✓	71	71	71	70	7 060	7 070	7 100	7 040
aflow30a			✓	27	27	27	27	718	736	753	718
aflow40b	✓		✓	900	900	898	894	3 868	3 879	3 861	3 725
air03			✓	2	3	3	3	63	96	83	94
air04	✓		✓	27	26	27	27	262	222	252	252
air05			✓	18	18	19	17	172	206	193	166
aligninq			✓	16	16	16	16	380	370	380	370
app1-2	✓	✓		5 326	4 415	7 200	7 200	177 568	135 961	309 051	720 001
arki001		✓		7 200	6 777	7 200	7 200	392	416	434	431
ash608gpia-3col	✓		✓	61	61	61	61	6 086	6 074	6 064	6 097
atlanta-ip				7 200	7 200	7 200	7 200	46 308	51 595	42 982	65 872
atm20-100				7 200	7 200	7 200	7 200	720 000	720 000	720 000	720 000
b2c1s1				7 200	7 200	7 200	7 200	17 792	9 487	10 847	24 912
bab1				7 200	7 200	7 200	7 200	1 650	1 590	4 326	3 238
bab3				7 200	7 200	7 200	7 200	720 271	720 122	720 545	720 325
bab5	✓	✓		677	871	1 116	2 819	2 746	2 707	2 737	2 896
bc			✓	1 146	1 151	1 145	1 143	15 356	15 685	15 301	15 302
bc1			✓	254	254	251	247	6 265	6 306	6 195	6 165
beasleyC3	✓	✓		99	76	108	72	532	504	515	556
bell3a			✓	8	8	8	8	0	0	0	0
bell5			✓	1	1	1	1	10	10	10	0
berlin_5_8_0				7 200	7 200	7 200	7 200	1 223	1 232	1 408	1 232
bg512142				7 200	7 200	7 200	7 200	36 444	38 919	33 846	37 986
biella1	✓	✓		722	722	234	709	2 495	2 521	1995	2 418
bienst1			✓	35	36	36	35	151	170	179	149
bienst2	✓		✓	342	342	343	343	471	467	500	474
binkar10_1	✓	✓		6	48	50	49	49	33	52	37
blend2		✓		2	2	2	5	19	30	29	217
bley_xl1	✓		✓	177	176	178	177	16 733	16 728	16 833	16 795
blp-ar98				7 200	7 200	7 200	7 200	720 000	720 000	720 002	720 001
blp-ic97				7 200	7 200	7 200	7 200	3 661	4 616	4 613	4 602
bnatt350	✓		✓	1 093	1 094	1 095	1 106	109 300	109 400	109 500	110 600
bnatt400			✓	1 302	1 296	1 300	1 307	101 100	100 700	101 000	101 400
buildingenergy				7 200	7 200	7 200	7 200	17 865	17 868	17 894	17 887
cap6000		✓		5	5	5	5	29	66	59	29
circ10-3				7 200	7 200	7 200	7 200	720 002	720 001	720 001	720 001
co-100				7 200	7 200	7 200	7 200	55 008	54 391	54 275	54 226
core2536-691	✓	✓		59	49	58	108	273	249	273	357
core4872-1529				7 200	7 200	7 200	7 200	18 330	19 309	18 893	19981
cov1075	✓		✓	135	135	135	135	166	166	167	166
csched007			✓	2 802	2 748	2 749	2 740	5 341	5 243	5 239	5 204
csched008			✓	446	446	446	446	1 454	1 455	1 464	1 424
csched010	✓		✓	2 567	2 569	2 569	2 567	4 819	4 829	4 832	4 817
d10200				7 200	7 200	7 200	7 200	750	751	751	751
d20200				7 200	7 200	7 200	7 200	1 127	1 124	1 121	1 121
dano3_3		✓		69	54	69	68	535	577	535	535
dano3_4			✓	82	62	82	37	457	437	477	477
dano3_5		✓		158	166	183	159	376	399	376	396
dano3mip				7 200	7 200	7 200	7 200	27 771	27 675	27 772	27 754
danoint	✓	✓		5 846	5 841	5 847	5 957	584	585	759	598
datt256				7 200	7 200	7 200	7 200	719 973	719 952	719 990	720 043
dc1c				7 200	7 200	7 200	7 200	11 541	10 906	27 174	11 459
dc1l				7 200	7 200	7 200	7 200	43 512	43 428	45 520	43 059
dcmulti		✓		3	2	3	2	15	14	15	14
dfn-gwin-UUM	✓	✓		76	68	68	88	156	166	166	169
dg012142				7 200	7 200	7 200	7 200	175 566	111 040	161 012	187 792
disctom			✓	2	2	2	2	200	160	200	160
dolom1				7 200	7 200	7 200	7 200	549 206	571 878	549 027	548 652
ds				7 200	7 200	7 200	7 200	541 837	542 147	541 870	541 299
ds-big				7 200	7 200	7 200	7 200	510 983	503 225	454 460	498 603
dsbmip			✓	1	1	1	1	43	43	43	83

continued on next page ...

ProblemName	2010	Diff	Equal	Time (sec.)				Prim. Int.			
				Exp.3	UCB	ε-greedy	ALNS off	Exp.3	UCB	ε-greedy	ALNS off
egout			✓	1	1	1	1	0	0	0	0
eil33-2	✓		✓	113	113	116	113	634	649	665	611
eilA101-2				7 200	7 200	7 200	7 200	89 667	90 110	89 880	90 164
eilB101	✓	✓		286	364	288	286	703	1 090	730	700
enigma			✓	1	1	1	1	30	70	60	30
enlight13	✓		✓	1	1	1	1	0	0	0	0
enlight14	✓		✓	1	1	1	1	1	3	2	3
enlight15			✓	1	1	1	1	0	10	10	10
enlight16			✓	1	1	1	1	3	7	4	2
enlight9			✓	1	1	1	1	1	0	1	0
ex10			✓	506	515	504	515	50 600	51 500	50 400	51 500
ex1010-pi				7 200	7 200	7 200	7 200	20 758	20 757	20 755	20 755
ex9	✓		✓	39	41	39	39	3 910	4 130	3 930	3 920
f2000				7 200	7 200	7 200	7 200	720 001	720 001	720 001	720 001
fast0507			✓	57	55	57	54	245	234	237	244
fiball				7 200	7 200	7 200	7 200	15 841	15 826	15 984	15 813
fiber			✓	3	3	3	3	26	24	24	42
fixnet6			✓	8	8	8	8	93	94	98	95
flugpl			✓	1	1	1	1	1	1	1	1
g200x740i				7 200	7 200	7 200	7 200	43 809	29 034	20 994	29 123
gen			✓	1	1	1	1	10	0	10	0
germanrr				7 200	7 200	7 200	7 200	58 915	51 678	59 013	59 020
germany50-DBM				7 200	7 200	7 200	7 200	6 482	4 321	6 599	5 701
gesa2			✓	1	1	1	1	88	44	79	44
gesa2-o			✓	1	1	1	1	22	52	44	22
gesa3			✓	5	5	5	5	12	10	20	30
gesa3_o			✓	3	3	3	3	60	90	90	90
glass4	✓	✓		1 526	1 264	1 710	3 358	36 139	28 260	42 952	80 752
gmu-35-40	✓			7 200	7 200	7 200	7 200	165	83	110	82
gmu-35-50				7 200	7 200	7 200	7 200	720 036	720 004	720 005	720 008
gmut-75-50				7 200	7 200	7 200	7 200	3 821	4 225	4 706	3 587
gmut-77-40				7 200	7 200	7 200	7 200	1 355	2 392	1 600	2 929
go19				7 200	7 200	7 200	7 200	293	296	293	293
gt2			✓	1	1	1	1	9	9	9	9
hanoi5				7 200	7 200	7 200	7 200	720 000	720 000	720 000	720 000
haprp			✓	1	1	1	1	20	20	22	20
ic97_potential				7 200	7 200	7 200	7 200	784	384	1 108	1 234
iis-100-0-cov	✓		✓	1 404	1 404	1 394	1 406	257	258	256	257
iis-bupa-cov	✓		✓	6 020	6 024	6 002	6 019	1 004	1 001	1 006	1 002
iis-pima-cov	✓		✓	810	805	806	806	1 111	1 107	1 089	1 088
in				7 189	7 188	7 189	7 188	470 211	470 156	470 244	470 192
ivu06-big				7 200	7 200	7 200	7 200	26 051	26 051	26 051	26 151
ivu52				7 200	7 200	7 200	7 200	107 599	114 156	118 778	108 599
janos-us-DDM				7 200	7 200	7 200	7 200	280	282	288	280
k16x240		✓		2 347	2 962	2 340	2 977	62	129	63	128
khb05250			✓	1	1	1	1	2	4	4	3
l152lav		✓		2	2	2	2	23	43	44	22
lectsched-1			✓	109	109	108	109	10 900	10 900	10 800	10 900
lectsched-1-obj				7 200	7 200	7 200	7 200	295 659	294 531	295 109	293 653
lectsched-2			✓	28	28	28	28	2 830	2 780	2 770	2 760
lectsched-3			✓	80	80	80	79	8 010	7 970	8 010	7 920
lectsched-4-obj	✓		✓	96	97	96	96	6 573	6 618	6 536	6 552
leo1		✓		4 498	4 500	3 445	4 485	4 319	4 318	3 575	4 303
leo2				7 200	7 200	7 200	7 200	15 384	15 996	21 447	16 901
liu				7 200	7 200	7 200	7 200	88 823	117 481	119 240	154 173
lotsize				7 200	7 200	7 200	7 200	14 444	14 447	14 347	14 443
lrn		✓		898	960	1 019	807	4 526	4 551	4 557	4 538
lrsa120				7 200	7 200	7 200	7 200	720 001	720 001	720 001	720 000
lseu			✓	1	1	1	1	8	16	15	8
m100n500k4r1	✓			7 200	7 200	7 200	7 200	29 184	29 184	29 190	29 183
macrophage	✓		✓	183	183	183	184	431	425	424	431
manna81			✓	1	1	1	1	13	12	13	25
map06		✓		726	778	822	1 094	7 632	7 315	9 890	19 657
map10		✓		994	982	958	922	12 046	11 764	11 677	11 623
map14			✓	807	808	825	806	3 350	3 350	3 354	3 350
map18	✓		✓	346	337	351	337	1 681	1 671	1 676	1 686
map20	✓		✓	235	237	247	234	1 411	1 409	1 410	1 406
markshare1				7 200	7 200	7 200	7 200	492 445	605 874	605 909	605 706
markshare2				7 200	7 200	7 200	7 200	685 128	685 178	686 991	685 154
markshare_5_0				7 200	7 200	7 200	7 200	397 313	397 155	397 343	397 371
mas74			✓	1 878	1 891	1 895	1 883	315	317	315	315
mas76			✓	165	163	164	164	14	7	7	14
maxgasflow				7 200	7 200	7 200	7 200	12 821	12 754	12 769	12 757
mc11		✓		407	408	408	442	690	696	685	695
mcsched	✓		✓	281	279	281	282	158	155	158	172
methanosarcina				7 200	7 200	7 200	7 200	15 003	15 013	14 974	15 046
mik-250-1-100-1	✓		✓	95	95	95	96	42	48	50	26
mine-166-5	✓		✓	84	84	85	83	4 150	4 142	4 192	4 093
mine-90-10	✓	✓		434	503	465	510	2 855	2 856	2 779	2 846
mining				7 200	7 200	7 200	7 200	774 420	771 961	770 608	772 441
misc03			✓	1	1	1	1	25	10	25	10
misc06			✓	1	1	1	1	0	0	10	10
misc07			✓	8	8	8	8	91	71	111	111
mitre			✓	15	16	16	16	1 490	1 580	1 560	1 570
mkc				7 200	7 200	7 200	7 200	1 058	1 059	1 464	1 062
mkc1		✓		56	33	47	35	51	87	48	90
mod008			✓	1	1	1	1	0	0	0	0
mod010		✓		1	1	1	1	53	21	53	21
mod011		✓		202	190	196	190	2 017	1 994	1 988	2 028
modglob			✓	1	1	1	1	0	0	0	0
momentum1				7 200	7 200	7 200	7 200	104 675	121 615	290 800	489 700
momentum3				7 200	7 200	7 200	7 200	441 021	337 153	441 953	443 549
msc98-ip	✓	✓		787	2 873	901	726	7 720	21 397	8 033	10 394

continued on next page …

ProblemName	2010	Diff	Equal	Time (sec.)				Prim. Int.			
				Exp.3	UCB	ε-greedy	ALNS off	Exp.3	UCB	ε-greedy	ALNS off
mspp16	✓		✓	711	711	712	701	28 600	28 800	28 500	28 400
mzzv11	✓		✓	191	183	186	180	7 483	7 349	7 434	7 216
mzzv42z		✓		98	104	147	97	7 325	7 613	7 467	7 275
n15-3				7 200	7 200	7 200	7 200	148 036	157 588	160 244	187 870
n3-3				7 200	7 200	7 200	7 200	17 848	18 766	18 114	21 282
n3700				7 200	7 200	7 200	7 200	35 056	35 047	33 779	47 682
n3705				7 200	7 200	7 200	7 200	48 923	48 919	40 248	48 984
n370a				7 200	7 200	7 200	7 200	39 635	39 760	39 649	39 629
n3div36	✓		✓	7 153	7 064	7 140	7 158	5 566	5 569	5 568	5 573
n3seq24	✓			7 200	7 200	7 200	7 200	42 413	41 152	41 927	41 988
n4-3	✓	✓		116	129	135	135	591	671	635	658
n9-3				7 200	7 200	7 200	7 200	9 717	8 942	2 574	6 976
nag				7 200	7 200	7 200	7 200	163 774	163 830	163 781	163 040
neos-1053234			✓	128	127	129	127	30	10	20	10
neos-1053591		✓		4	5	5	5	21	11	22	11
neos-1056905				7 200	7 200	7 200	7 200	52 523	65 360	88 185	69 906
neos-1058477			✓	3	3	3	3	10	10	10	20
neos-1061020		✓		3 777	3 621	3 779	3 775	8 252	8 171	8 240	8 251
neos-1062641			✓	1	1	1	1	10	30	30	30
neos-1067731				7 200	7 200	7 200	7 200	1 002	1 001	1 002	1 001
neos-1096528		✓		4 563	4 343	4 542	4 537	57 648	56 763	57 428	57 611
neos-1109824	✓		✓	17	17	17	17	625	635	605	606
neos-1120495			✓	6	6	6	6	542	553	562	523
neos-1121679				7 200	7 200	7 200	7 200	203 015	263 262	263 381	264 242
neos-1122047			✓	7	7	7	7	690	690	680	700
neos-1126860		✓		1 803	1 781	1 833	1 792	1 977	1 959	1 974	4 494
neos-1151496			✓	4	4	4	4	370	420	400	410
neos-1171448		✓		52	51	6	51	320	311	282	319
neos-1171692		✓		28	57	24	21	157	142	137	151
neos-1171737		✓		1 879	6 810	7 200	3 075	2 390	10 238	10038	6 054
neos-1173026			✓	1	1	1	1	19	19	15	15
neos-1200887			✓	18	18	18	19	30	28	35	46
neos-1208069			✓	39	39	39	39	3 900	3 880	3 920	3 855
neos-1208135			✓	37	36	37	36	3 550	3 470	3 570	3 460
neos-1211578			✓	2	2	2	2	8	25	25	25
neos-1215259		✓		46	43	38	37	1 408	1 365	1 343	1 339
neos-1215891			✓	74	74	75	75	741	771	761	741
neos-1223462			✓	259	259	259	257	25 900	25 900	25 900	25 600
neos-1224597			✓	2	2	2	2	160	190	200	160
neos-1228986			✓	12	12	12	12	17	9	17	5
neos-1281048			✓	9	9	9	10	312	312	312	352
neos-1311124				7 200	7 200	7 200	7 200	17	16	16	14
neos-1324574			✓	2 748	2 715	2 717	2 755	20	20	20	20
neos-1330346				7 200	7 200	7 200	7 200	10	10	10	10
neos-1330635			✓	1	1	1	1	28	51	48	51
neos-1337307	✓			7 200	7 200	7 200	7 200	7 977	7 987	7 957	7 967
neos-1346382				7 200	7 200	7 200	7 200	29	19	19	31
neos-1354092				7 200	7 200	7 200	7 200	720 000	720 000	720 000	720 000
neos-1367061			✓	18	18	18	17	986	986	986	986
neos-1396125	✓		✓	78	78	78	78	1 605	1 577	1 582	1 570
neos-1407044				7 200	7 200	7 200	7 200	720 001	720 000	720 000	720 001
neos-1413153			✓	5	5	5	4	203	229	237	197
neos-1415183			✓	8	8	8	8	521	489	523	489
neos-1417043			✓	10	10	10	10	990	980	990	980
neos-1420205			✓	3	3	3	3	5	5	5	19
neos-1420546				7 200	7 200	7 200	7 200	2 835	2 848	2 829	2 829
neos-1420790				7 200	7 200	7 200	7 200	4 985	4 952	4 948	4 950
neos-1423785				7 200	7 200	7 200	7 200	138 760	152 139	153 211	196 728
neos-1425699			✓	1	1	1	1	0	0	0	0
neos-1426662				7 200	7 200	7 200	7 200	70	82	64	101
neos-1427181				7 200	7 200	7 200	7 200	20	25	20	20
neos-1427261				7 200	7 200	7 200	7 200	422	298	284	424
neos-1429185				7 200	7 200	7 200	7 200	48	29	29	46
neos-1429212				7 200	7 200	7 200	7 200	720 003	720 001	720 001	720 002
neos-1429461				7 200	7 200	7 200	7 200	25	25	25	24
neos-1430701			✓	13	12	13	13	13	13	28	31
neos-1430811				7 200	7 200	7 200	7 200	366 219	356 555	376 302	476 356
neos-1436709				7 200	7 200	7 200	7 200	22	22	22	23
neos-1436713				7 200	7 200	7 200	7 200	266	256	260	259
neos-1437164			✓	1	1	1	1	62	91	102	61
neos-1439395			✓	1 900	1 903	1 905	1 898	31	17	20	17
neos-1440225			✓	77	77	78	77	7 740	7 730	7 760	7 740
neos-1440447			✓	5	6	6	5	15	23	22	23
neos-1440457				7 200	7 200	7 200	7 200	19	18	23	18
neos-1440460				7 200	7 200	7 200	7 200	23	19	21	22
neos-1441553		✓		2	2	2	2	196	156	204	174
neos-1442119				7 200	7 200	7 200	7 200	18	21	21	20
neos-1442657				7 200	7 200	7 200	7 200	16	16	15	16
neos-1445532				7 200	7 200	7 200	7 200	9 821	9 845	9 884	9 841
neos-1445738				7 200	7 200	7 200	7 200	62 078	69 034	69 069	69 707
neos-1445743		✓		41	55	56	55	2 478	2 463	2 497	2 456
neos-1445755		✓		45	62	60	62	2 629	2 540	2 511	2 535
neos-1445765		✓		35	35	36	36	2 520	2 555	2 498	2 502
neos-1451294			✓	1 424	1 427	1 441	1 424	48 283	48 511	48 943	48 284
neos-1456979			✓	2 839	2 845	2 870	2 836	7 720	7 705	7 876	7 658
neos-1460246				7 200	7 200	7 200	7 200	23	23	23	23
neos-1460265			✓	1	1	1	1	91	82	51	51
neos-1460543				7 200	7 200	7 200	7 200	9 423	9 422	9 425	9 425
neos-1460641				7 200	7 200	7 200	7 200	101	142	127	142
neos-1461051			✓	22	22	22	22	2 247	2 233	2 238	2 245
neos-1464762				7 200	7 200	7 200	7 200	150	150	150	150
neos-1467067				7 200	7 200	7 200	7 200	18	15	23	18
neos-1467371				7 200	7 200	7 200	7 200	1 528	2 242	1 530	1 527

continued on next page . . .

ProblemName	2010	Diff	Equal	Time (sec.)				Prim. Int.			
				Exp.3	UCB	ε-greedy	ALNS off	Exp.3	UCB	ε-greedy	ALNS off
neos-1467467				7200	7200	7200	7200	3623	3753	3867	4759
neos-1480121			✓	1	1	1	1	3	4	3	8
neos-1489999			✓	2	3	2	2	15	29	27	27
neos-1516309			✓	1	1	1	1	42	82	82	41
neos-1582420			✓	19	19	19	19	401	401	431	431
neos-1593097				7200	7200	7200	7200	1524	370	1369	2153
neos-1595230				7200	7200	7200	7200	71	70	65	71
neos-1597104			✓	594	596	595	595	49694	49857	49783	49755
neos-1599274			✓	1	1	1	1	53	43	53	33
neos-1601936	✓		✓	1104	1104	1114	1092	107279	107297	108239	106136
neos-1603512			✓	2	2	2	2	250	240	250	240
neos-1603518			✓	10	10	10	10	990	980	990	1030
neos-1603965				7200	7200	7200	7200	1240	1262	1267	1272
neos-1605061				7200	7200	7200	7200	632446	632569	632550	632390
neos-1605075		✓		4915	4857	7200	4850	476171	470463	548721	469782
neos-1616732			✓	7128	7110	7159	7129	388	387	390	390
neos-1620770				7200	7200	7200	7200	158	158	158	158
neos-1620807			✓	6	6	6	6	0	10	10	10
neos-1622252				7200	7200	7200	7200	21	21	61	52
neos-430149			✓	35	35	35	35	418	452	411	465
neos-476283	✓	✓		346	353	343	310	1567	1567	1579	1577
neos-480878		✓		77	52	53	121	44	53	21	24
neos-494568		✓		4	4	3	9	114	84	91	124
neos-495307				7200	7200	7200	7200	26	26	26	23
neos-498623		✓		8	9	9	9	368	381	326	372
neos-501453			✓	1	1	1	1	10	10	10	10
neos-501474			✓	1	1	1	1	10	10	10	10
neos-503737			✓	202	200	201	199	1086	1065	1095	1088
neos-504674			✓	77	76	76	76	1029	995	1000	999
neos-504815		✓		24	21	23	21	600	564	574	578
neos-506422		✓		10	24	15	14	1050	2430	1500	1450
neos-506428				7200	7200	7200	7200	215960	216040	215480	215787
neos-512201		✓		34	36	38	35	1088	1115	1057	1096
neos-522351			✓	2	2	2	2	83	97	102	83
neos-525149			✓	2	2	2	2	160	151	151	152
neos-530627			✓	1	1	1	1	0	0	0	0
neos-538867			✓	61	61	61	61	360	379	396	359
neos-538916			✓	68	68	68	67	1131	1135	1110	1092
neos-544324			✓	25	25	25	25	589	588	588	586
neos-547911			✓	21	22	22	21	264	273	270	263
neos-548047			✓	5865	5902	5898	5907	30820	31008	30966	31079
neos-548251				7200	7200	7200	7200	75429	74303	85178	74055
neos-551991			✓	49	48	49	47	319	295	321	312
neos-555001			✓	1	1	1	1	57	57	52	87
neos-555298			✓	50	49	50	49	802	768	808	787
neos-555343		✓		5858	2049	2039	2048	5426	4605	4582	4602
neos-555424		✓		3327	4902	3658	4919	9595	11324	7116	11341
neos-555694		✓		5	3	3	4	152	119	149	133
neos-555771		✓		3	2	2	3	50	60	70	51
neos-555884				7200	7200	7200	7200	27176	27173	27173	27069
neos-555927				7200	7200	7200	7200	2758	2758	2756	2749
neos-565672				7200	7200	7200	7200	192119	192095	191394	191720
neos-565815			✓	8	8	8	8	140	140	110	120
neos-570431			✓	13	13	13	12	83	73	83	68
neos-574665				7200	7200	7200	7200	307	304	297	307
neos-578379			✓	205	206	205	206	20500	20600	20500	20600
neos-582605				7200	7200	7200	7200	1200	1200	1200	1220
neos-583731			✓	16	17	17	17	1630	1650	1650	1660
neos-584146				7200	7200	7200	7200	0	0	0	0
neos-584851			✓	3	4	4	3	116	138	142	116
neos-584866				7200	7200	7200	7200	27892	27892	27928	27886
neos-585192			✓	28	28	28	28	619	605	579	584
neos-585467			✓	10	11	11	10	271	306	316	281
neos-593853		✓		50	281	71	653	120	179	121	395
neos-595904			✓	21	21	22	21	1280	1310	1350	1270
neos-595905			✓	6	5	5	5	270	210	260	210
neos-595925			✓	20	20	20	20	493	453	483	453
neos-598183		✓		13	11	11	13	137	140	140	157
neos-603073			✓	136	134	134	134	276	273	280	273
neos-611135				7200	7200	7200	7200	33923	51452	16642	25575
neos-611838		✓		18	17	18	17	26	55	55	25
neos-612125			✓	11	10	11	10	23	23	53	23
neos-612143			✓	25	23	23	21	25	45	25	25
neos-612162		✓		25	25	26	22	57	54	57	26
neos-619167		✓		2547	1724	1735	1730	27457	25220	25573	25295
neos-631164				7200	7200	7200	7200	9186	9184	9184	9186
neos-631517				7200	7200	7200	7200	2951	2953	2954	2954
neos-631694				7200	7200	7200	7200	2100	2160	2160	2150
neos-631709				7200	7200	7200	7200	4431	4881	4908	4932
neos-631710				7200	7200	7200	7200	50634	50730	50494	50563
neos-631784				7200	7200	7200	7200	1479	1490	1490	1509
neos-632335			✓	1	1	1	1	37	67	59	67
neos-633273			✓	1	1	1	1	27	60	50	28
neos-655508			✓	2	2	2	2	210	200	220	200
neos-662469		✓		6551	7028	7200	6507	12132	12432	12771	11605
neos-686190	✓		✓	143	142	142	141	1770	1764	1728	1723
neos-691058				7200	7200	7200	7200	188	191	188	189
neos-691073				7200	7200	7200	7200	147	147	147	147
neos-693347			✓	1404	1383	1376	1373	30563	28957	28998	28500
neos-702280				7200	7200	7200	7200	28586	29694	28239	27805
neos-709469			✓	1	1	1	1	60	60	60	100
neos-717614		✓		38	5	6	933	122	121	122	125
neos-738098				7200	7200	7200	7200	720001	720000	720000	720001

continued on next page . . .

ProblemName	2010	Diff	Equal	Time (sec.)				Prim. Int.			
				Exp.3	UCB	ε-greedy	ALNS off	Exp.3	UCB	ε-greedy	ALNS off
neos-775946			✓	11	11	11	11	538	557	567	540
neos-777800			✓	25	23	22	18	2 510	2 260	2 200	1 810
neos-780889			✓	48	47	48	48	4 800	4 740	4 830	4 770
neos-785899			✓	3	3	3	3	243	263	253	273
neos-785912			✓	10	10	10	10	1 012	976	972	1 012
neos-785914			✓	1	1	1	1	60	90	100	60
neos-787933			✓	2	2	2	2	188	188	188	188
neos-791021			✓	46	46	46	46	3 571	3 581	3 582	3 521
neos-796608			✓	1	1	1	1	0	0	0	0
neos-799838		✓		18	21	18	24	663	684	673	720
neos-801834			✓	23	23	22	22	77	78	74	76
neos-803219		✓		71	72	73	71	63	117	105	103
neos-803220		✓		134	131	132	145	133	131	122	126
neos-806323			✓	38	37	38	37	677	638	639	632
neos-807454			✓	2	2	2	2	164	164	166	196
neos-807456				7 200	7 200	7 200	7 200	720 000	720 000	720 000	720 000
neos-807639			✓	29	29	29	29	61	29	39	38
neos-807705			✓	42	43	42	42	355	385	394	364
neos-808072			✓	19	19	19	19	1 408	1 408	1 407	1 407
neos-808214			✓	10	10	8	8	990	980	830	830
neos-810286			✓	20	20	20	20	2 020	2 010	2 010	2 010
neos-810326			✓	37	38	37	37	1 242	1 248	1 257	1 236
neos-820146				7 200	7 200	7 200	7 200	720 000	720 000	720 000	720 000
neos-820157				7 200	7 200	7 200	7 200	720 000	720 000	720 000	720 000
neos-820879			✓	54	54	51	53	1 209	1 189	1 151	1 176
neos-824661			✓	22	22	22	22	1 078	1 068	1 080	1 084
neos-824695			✓	7	7	7	7	332	341	334	334
neos-825075			✓	1	2	2	2	127	160	147	150
neos-826224			✓	7	7	8	7	382	362	392	375
neos-826250			✓	2	3	3	3	138	142	168	187
neos-826650			✓	1 742	1 738	1 738	1 739	19 725	19 615	19 615	19 525
neos-826694			✓	5	5	5	5	247	248	248	248
neos-826812			✓	2	2	2	3	143	117	113	156
neos-826841				7 200	7 200	7 200	7 200	50	50	50	50
neos-827015		✓		457	480	482	470	13 786	12 739	12 738	12 641
neos-827175			✓	3	3	3	3	216	236	236	207
neos-829552		✓		111	111	162	111	6 921	6 936	7 681	6 918
neos-830439			✓	1	1	1	1	20	30	30	10
neos-831188			✓	176	175	174	174	2 710	2 740	2 690	2 690
neos-839838			✓	1 601	1 594	1 598	1 601	620	621	624	628
neos-839859		✓		47	46	47	50	219	219	170	222
neos-839894				7 200	7 200	7 200	7 200	72 120	45 063	72 203	72 120
neos-841664				7 200	7 200	7 200	7 200	1 178	1 208	948	1 248
neos-847051				7 200	7 200	7 200	7 200	50	52	52	44
neos-847302				7 200	7 200	7 200	7 200	1 681	1 648	1 646	1 660
neos-848150			✓	1	1	1	1	60	60	60	60
neos-848198				7 200	7 200	7 200	7 200	17 584	17 969	17 833	21 044
neos-848589				7 200	7 200	7 200	7 200	85 111	74 440	74 276	84 883
neos-848845			✓	367	364	367	363	36 700	36 400	36 700	36 300
neos-849702	✓		✓	90	90	90	89	9 010	8 980	8 950	8 940
neos-850681		✓		5	3	7	5	139	114	167	143
neos-856059			✓	4 377	4 398	4 408	4 394	1 907	1 917	1 920	1 913
neos-859770			✓	158	158	157	158	15 758	15 835	15 706	15 820
neos-860244			✓	6	6	6	6	220	210	220	210
neos-860300			✓	20	20	20	20	887	890	892	885
neos-862348		✓		9	5	7	7	243	195	223	204
neos-863472			✓	58	56	56	57	51	51	51	51
neos-872648				7 200	7 200	7 200	7 200	3 932	4 461	15 298	15 134
neos-873061				7 200	7 200	7 200	7 200	12 466	5 199	8 936	12 106
neos-876808				7 200	7 200	7 200	7 200	322 179	281 595	356 148	359 589
neos-880324			✓	1	1	1	1	60	60	60	100
neos-881765			✓	1	1	1	1	10	20	20	20
neos-885086			✓	604	616	606	603	837	852	837	836
neos-885524		✓		3 758	7 200	2 326	7 200	34 332	46 770	34 624	46 169
neos-886822			✓	1 883	1 885	1 884	1 875	2 329	2 326	2 292	2 287
neos-892255			✓	22	22	22	22	30	30	30	10
neos-905856			✓	508	511	511	508	30 367	30 450	30 367	30 267
neos-906865			✓	234	235	234	232	77	74	67	50
neos-911880			✓	2 625	2 621	2 627	2 614	154	152	157	147
neos-911970			✓	7	7	7	7	112	112	113	112
neos-912015			✓	10	10	10	10	648	603	642	602
neos-912023			✓	4	4	4	4	420	390	400	420
neos-913984			✓	12	11	12	11	1 190	1 130	1 170	1 130
neos-914441			✓	63	62	62	63	986	970	968	987
neos-916173			✓	138	139	138	139	2 289	2 292	2 328	2 340
neos-916792	✓	✓		1 260	1 229	1 241	1 253	4 912	4 992	5 032	5 137
neos-930752				7 200	7 200	7 200	7 200	7 584	5 332	7 665	7 560
neos-931517				7 200	7 200	7 200	7 200	50 251	57 152	57 410	63 640
neos-931538			✓	9	9	9	9	500	480	480	503
neos-932721		✓		8	2	8	8	86	105	85	95
neos-932816		✓		5 194	4 977	5 166	4 252	27 700	27 721	27 626	21 493
neos-933364			✓	326	327	326	325	49	51	55	49
neos-933550			✓	4	3	4	3	390	350	390	350
neos-933562				7 200	7 200	7 200	7 200	999	986	1 008	999
neos-933638		✓		104	97	105	104	3 001	2 950	3 031	3 001
neos-933815				7 200	7 200	7 200	7 200	65	74	65	74
neos-933966		✓		141	43	141	141	4 123	4 052	4 122	4 131
neos-934278	✓		✓	140	139	140	139	4 669	4 667	4 678	4 667
neos-934441				7 200	7 200	7 200	7 200	4 744	4 890	4 876	4 876
neos-934531			✓	265	267	264	266	26 500	26 700	26 400	26 600
neos-935234				7 200	7 200	7 200	7 200	5 284	6 070	6 189	6 248
neos-935348				7 200	7 200	7 200	7 200	6 551	8 185	8 007	8 185
neos-935496				7 200	7 200	7 200	7 200	3 030	2 913	3 005	4 746

continued on next page . . .

ProblemName	2010	Diff	Equal	Time (sec.)				Prim. Int.			
				Exp.3	UCB	ε-greedy	ALNS off	Exp.3	UCB	ε-greedy	ALNS off
neos-935627			✓	343	343	344	344	1 995	1 989	2 004	2 000
neos-935674				7 200	7 200	7 200	7 200	3 307	3 707	3 415	3 283
neos-935769			✓	51	51	51	51	1 716	1 716	1 723	1 718
neos-936660			✓	201	202	202	200	2 825	2 736	2 850	2 746
neos-937446		✓		142	141	181	142	3 675	3 666	3 491	3 663
neos-937511			✓	50	50	50	50	2 478	2 478	2 480	2 478
neos-937815				7 200	7 200	7 200	7 200	9 310	11 023	9 240	9 278
neos-941262				7 200	7 200	7 200	7 200	36 301	36 150	36 578	36 338
neos-941313		✓		297	372	379	371	8 089	8 537	8 577	8 525
neos-941698			✓	6	6	6	6	610	610	580	620
neos-941717				7 200	7 200	7 200	7 200	2 005	1 906	1 990	1 926
neos-941782				7 200	7 200	7 200	7 200	2 095	2 098	2 107	2 094
neos-942323			✓	3	3	3	3	150	151	151	190
neos-942830		✓		1 563	1 560	730	1 559	1 633	2 057	1 575	2 039
neos-942886			✓	1	1	1	1	17	48	45	46
neos-948126				7 200	7 200	7 200	7 200	62 399	62 449	62 396	62 354
neos-948268			✓	5	5	5	5	470	490	470	450
neos-948346				7 200	7 200	7 200	7 200	27 597	34 850	35 311	33 226
neos-950242			✓	494	492	493	492	33 080	32 960	32 980	32 880
neos-952987				7 200	7 200	7 200	7 200	720 003	720 121	720 001	720 000
neos-953928		✓		48	54	180	54	1 301	1 291	1 324	1 301
neos-954925		✓		431	633	325	588	6 602	6 409	6 455	6 458
neos-955215			✓	3 452	3 416	3 409	3 402	35	35	56	35
neos-955800		✓		71	72	72	71	3 933	4 425	4 454	4 405
neos-956971		✓		251	228	424	476	3 455	3 463	3 470	3 455
neos-957143		✓		174	59	170	174	3 296	3 299	3 299	3 316
neos-957270			✓	3	3	3	3	280	280	290	280
neos-957323		✓		56	40	39	37	851	842	851	849
neos-957389			✓	18	18	17	17	1 740	1 740	1 710	1 710
neos-960392		✓		132	119	130	129	3 780	3 616	3 786	3 774
neos-983171				7 200	7 200	7 200	7 200	33 111	32 832	35 171	34 935
neos-984165				7 200	7 200	7 200	7 200	74 073	85 280	104 587	81 329
neos13	✓		✓	390	388	391	384	5 682	5 730	5 733	5 593
neos15		✓		4 334	7 200	4 425	6 181	454	759	461	614
neos16				7 200	7 200	7 200	7 200	2 174	2 184	2 154	2 153
neos18	✓		✓	56	56	56	56	383	372	380	359
neos6		✓		62	110	74	73	2 052	3 055	4 306	4 244
neos788725			✓	253	251	254	253	25 308	25 079	25 369	25 272
neos808444			✓	3 100	3 107	3 086	3 119	310 000	310 700	308 600	311 900
neos858960			✓	2 377	2 371	2 381	2 381	237 667	237 079	238 050	238 099
net12	✓		✓	991	984	989	987	7 754	7 754	7 754	7 745
netdiversion	✓	✓		698	467	693	465	45 150	41 898	44 710	41 798
newdano	✓			7 200	7 200	7 200	7 200	2 696	2 633	2 652	2 655
nobel-eu-DBE				7 200	7 200	7 200	7 200	720 000	720 000	720 000	720 000
noswot	✓	✓		125	129	125	129	9	10	34	21
ns1111636				7 200	7 200	7 200	7 200	74 578	74 570	74 555	74 580
ns1116954				7 200	7 200	7 200	7 200	720 002	720 003	720 003	720 002
ns1158817			✓	247	247	246	246	24 672	24 657	24 594	24 641
ns1208400	✓		✓	364	364	366	361	36 100	36 100	36 300	35 700
ns1456591				7 200	7 200	7 200	7 200	244 990	245 058	246 515	245 984
ns1606230			✓	6 328	6 325	6 333	6 323	506 541	506 275	507 128	506 152
ns1631475				7 200	7 200	7 200	7 200	267 598	267 815	266 851	267 707
ns1644855			✓	986	986	980	985	98 555	98 643	98 042	98 500
ns1663818				7 194	7 194	7 195	7 195	719 436	719 421	719 467	719 459
ns1685374				7 200	7 200	7 200	7 200	720 001	720 001	720 000	720 001
ns1686196			✓	13	13	13	13	1 278	1 329	1 294	1 314
ns1688347	✓		✓	150	152	152	148	3 636	3 655	3 646	3 590
ns1696083				7 200	7 200	7 200	7 200	701 703	705 076	702 220	703 518
ns1702808			✓	39	38	38	38	3 860	3 801	3 826	3 810
ns1745726			✓	14	15	15	15	1 427	1 463	1 498	1 521
ns1758913	✓		✓	3 267	3 266	3 266	3 251	235 016	234 846	234 946	233 825
ns1766074	✓		✓	589	585	587	586	58 928	58 456	58 695	58 591
ns1769397			✓	23	23	22	23	2 289	2 281	2 246	2 349
ns1778858				7 200	7 200	7 200	7 200	720 000	720 000	720 000	720 000
ns1830653	✓		✓	218	219	219	219	4 536	4 568	4 545	4 546
ns1853823				7 200	7 200	7 200	7 200	699 473	699 473	699 472	699 474
ns1854840				7 200	7 200	7 200	7 200	683 005	683 005	683 005	683 005
ns1856153				7 200	7 200	7 200	7 200	250 968	250 899	250 897	250 905
ns1904248				7 200	7 200	7 200	7 200	109 303	109 302	109 304	109 295
ns1905797				7 200	7 200	7 200	7 200	720 000	720 000	720 000	720 000
ns1905800				7 200	7 200	7 200	7 200	720 001	720 001	720 000	720 000
ns1952667			✓	1 317	1 286	1 288	1 272	131 700	128 600	128 800	127 200
ns2081729				7 200	7 200	7 200	7 200	720 001	720 003	720 007	720 003
ns2118727			✓	831	833	835	834	83 061	83 270	83 480	83 429
ns2124243				7 200	7 200	7 200	7 200	50 743	50 744	50 872	50 642
ns2137859				7 200	7 200	7 200	7 200	131 841	118 770	132 197	131 600
ns4-pr3				7 200	7 200	7 200	7 200	124	102	65	103
ns4-pr9				7 200	7 200	7 200	7 200	28	18	35	44
ns894236				7 200	7 200	7 200	7 200	173 111	173 011	175 517	172 711
ns894244			✓	1 435	1 441	1 438	1 438	143 500	144 100	143 800	143 800
ns894786				7 200	7 200	7 200	7 200	181 800	182 000	181 500	180 400
ns894788				7 200	7 200	7 200	7 200	720 001	720 000	720 000	720 000
ns903616				7 200	7 200	7 200	7 200	720 001	720 001	720 000	720 000
ns930473				7 200	7 200	7 200	7 200	89 399	89 457	89 559	89 425
nsa			✓	3	3	3	3	292	243	282	242
nsr8k				7 200	7 200	7 200	7 200	522 791	404 722	523 587	511 789
nsrand-ipx			✓	390	390	400	389	1 546	1 544	1 567	1 545
nu120-pr3				7 200	7 200	7 200	7 200	36 343	36 190	36 351	36 229
nu60-pr9				7 200	7 200	7 200	7 200	2 970	2 978	1 428	2 994
nug08			✓	124	123	130	129	2 699	2 693	2 834	2 768
nw04		✓		33	48	36	34	1 403	1 611	1 415	1 395
opm2-z10-s2				7 200	7 200	7 200	7 200	127 518	98 336	98 440	127 029
opm2-z11-s8				7 200	7 200	7 200	7 200	91 206	91 405	90 793	91 295

continued on next page . . .

ProblemName	2010	Diff	Equal	Time (sec.)				Prim. Int.			
				Exp.3	UCB	ε-greedy	ALNS off	Exp.3	UCB	ε-greedy	ALNS off
opm2-z12-s14				7200	7200	7200	7200	180837	181287	181324	181653
opm2-z12-s7				7200	7200	7200	7200	281122	280743	281300	281374
opm2-z7-s2	✓	✓		430	727	489	487	7305	4967	7470	8181
opt1217			✓	1	1	1	1	0	0	0	10
p0033			✓	1	1	1	1	1	1	1	10
p0201			✓	1	1	1	1	12	31	30	14
p0282			✓	1	1	1	1	7	3	5	1
p0548			✓	1	1	1	1	8	10	10	15
p100x588b				7200	7200	7200	7200	6993	13682	12270	12265
p2756		✓		4	4	4	4	63	83	92	59
p2m2p1m1p0n100				7200	7200	7200	7200	720000	720000	720000	720000
p6b				7200	7200	7200	7200	1687	1687	1691	1694
p80x400b				7200	7200	7200	7200	720000	720001	720027	720001
pb-simp-nonunif				7200	7200	7200	7200	346806	346826	346973	346753
pg		✓		69	67	57	67	383	358	358	375
pg5_34	✓	✓		1217	1834	951	1026	129	182	105	118
pigeon-10	✓		✓	594	588	589	586	110	110	110	110
pigeon-11			✓	3752	3760	3741	3751	100	100	140	90
pigeon-12				7200	7200	7200	7200	264	296	271	270
pigeon-13				7200	7200	7200	7200	210	210	210	210
pigeon-19				7200	7200	7200	7200	847	914	836	940
pk1			✓	174	174	174	174	1146	1147	1166	1181
pp08a		✓		2	2	2	2	51	31	50	34
pp08aCUTS		✓		2	2	2	2	37	38	37	52
probportfolio				7200	7200	7200	7200	65405	59265	60348	60345
prod1			✓	24	24	24	25	20	17	20	23
prod2			✓	196	196	195	196	929	920	918	931
protfold				7200	7200	7200	7200	144898	129453	111179	129346
pw-myciel4	✓		✓	3756	3746	3744	3741	2419	2324	2316	2324
qap10			✓	67	66	68	67	1561	1534	1609	1571
qiu	✓	✓		33	34	33	34	1275	1399	1255	1429
qnet1			✓	3	3	3	3	56	88	80	88
qnet1_o		✓		1	2	1	2	23	54	40	38
queens-30				7200	7200	7200	7200	23032	23114	22985	23002
r80x800				7200	7200	7200	7200	227	428	323	442
rail01			✓	7032	7035	7032	7033	703243	703478	703182	703295
rail02				7200	7200	7200	7200	254304	256166	254889	253207
rail03				7200	7199	7200	7200	719960	719918	719950	719966
rail507	✓		✓	71	62	63	60	378	372	374	371
ramos3				7200	7200	7200	7200	116041	116043	116042	116041
ran14x18		✓		2384	1694	2139	2318	756	613	674	726
ran14x18-disj-8			✓	2086	2066	2070	2070	592	581	585	586
ran14x18_1			✓	3501	3473	3467	3473	731	725	724	723
ran16x16	✓	✓		62	76	62	61	53	81	51	68
rd-rplusc-21				7200	7200	7200	7200	14235	14234	14239	14238
reblock166		✓		7200	7200	4715	7200	20103	20111	21157	19943
reblock354				7200	7200	7200	7200	5344	5108	4888	5595
reblock420				7200	7200	7200	7200	59084	59281	59189	58775
reblock67	✓		✓	235	235	235	234	1592	1583	1570	1560
rentacar		✓		2	1	2	2	154	76	158	127
rgn			✓	1	1	1	1	23	23	23	46
rlp1				7200	7200	7200	7200	4	4	4	3
rmatr100-p10	✓		✓	89	89	89	89	343	351	332	332
rmatr100-p5	✓		✓	94	85	86	85	367	376	387	385
rmatr200-p10				7200	7200	7200	7200	24021	26472	32304	21643
rmatr200-p20				7200	7200	7200	7200	15005	19571	12388	15010
rmatr200-p5				7200	7200	7200	7200	15222	19624	7108	16546
rmine10				7200	7200	7200	7200	21535	21537	21455	21534
rmine14				7200	7200	7200	7200	605761	605764	605743	605728
rmine21				7199	7198	7198	7198	368578	368459	368619	368352
rmine25				7196	7197	7197	7197	673407	673451	673477	673407
rmine6	✓		✓	1129	1122	1120	1118	1130	1121	1124	1120
rocII-4-11	✓		✓	207	208	217	211	5269	5292	5467	5342
rocII-7-11				7200	7200	7200	7200	259280	259139	259269	259070
rocII-9-11				7200	7200	7200	7200	328847	331902	328876	327922
rococoB10-011000				7200	7200	7200	7200	5171	5246	5157	5165
rococoC10-001000	✓	✓		443	411	397	442	1439	1630	1179	1432
rococoC11-011100				7200	7200	7200	7200	21247	21536	21147	21251
rococoC12-111000				7200	7200	7200	7200	49407	47670	55330	55328
roll3000	✓		✓	46	46	46	46	393	393	393	393
rout		✓		121	137	135	135	101	94	81	82
roy			✓	1	1	1	1	7	7	7	10
rvb-sub				7200	7200	7200	7200	151881	151891	398547	151580
satellites1-25	✓		✓	620	604	615	609	58700	57200	58200	57700
satellites2-60-fs				7200	7200	7200	7200	720000	720001	720001	720000
sct1				7200	7200	7200	7200	69014	80533	54244	53583
sct32				7200	7200	7200	7200	272300	272110	271956	270994
sct5				7200	7200	7200	7200	21924	53153	19131	18951
set1ch			✓	1	1	1	1	13	28	27	28
set3-10				7200	7200	7200	7200	147327	120736	69623	69541
set3-15				7200	7200	7200	7200	109638	68669	73577	101371
set3-20				7200	7200	7200	7200	64192	97008	66692	86169
seymour				7200	7200	7200	7200	1588	1742	1438	1264
seymour-disj-10				7200	7200	7200	7200	1906	2449	2339	1908
shipsched				7200	7200	7200	7200	294258	294107	294053	294241
shs1023				7200	7200	7200	7200	533971	534558	532187	530793
siena1				7200	7200	7200	7200	501518	180502	300985	176029
sing161				7200	7200	7200	7200	119015	118531	118168	117614
sing2				7200	7200	7200	7200	34156	37086	40746	40694
sing245				7200	7200	7200	7200	93315	93181	90179	92871
sing359				7200	7200	7200	7200	199281	198945	199536	199313
sp97ar		✓		4965	5119	5508	6092	3585	4022	4330	3962
sp97ic				7200	7200	7200	7200	1953	1600	1291	946

continued on next page ...

ProblemName	2010	Diff	Equal	Time (sec.)				Prim. Int.			
				Exp.3	UCB	ε-greedy	ALNS off	Exp.3	UCB	ε-greedy	ALNS off
sp98ar				7200	7200	7200	7200	6469	4884	5810	4753
sp98ic	✓		✓	1434	1440	1441	1431	3189	3186	3188	3176
sp98ir	✓	✓		61	51	82	66	227	241	299	291
stein27			✓	1	2	1	1	0	0	0	0
stein45			✓	28	28	28	28	9	11	12	11
stockholm				7200	7200	7200	7200	73504	73507	73495	73496
stp3d				7200	7200	7200	7200	720003	720004	720002	720003
sts405				7200	7200	7200	7200	13499	13491	13487	13484
sts729				7200	7200	7200	7200	6714	6706	6727	6712
swath				7200	7200	7200	7200	4485	4502	4527	4457
t1717				7200	7200	7200	7200	182238	182192	182164	182199
t1722				7200	7200	7200	7200	97985	96676	91628	97322
tanglegram1	✓		✓	385	390	388	386	895	900	905	894
tanglegram2	✓		✓	9	9	9	9	177	169	169	195
timtab1	✓	✓		89	100	95	101	578	859	609	860
timtab2				7200	7200	7200	7200	14860	14888	14883	14852
toll-like				7200	7200	7200	7200	1835	1844	1839	1841
tr12-30			✓	1442	1444	1440	1442	61	60	61	59
triptim1	✓	✓		447	268	447	450	22037	21026	22043	22053
triptim2				7200	7200	7200	7200	173782	174882	177781	174282
triptim3				7200	7200	7200	7200	81753	81753	81753	81853
tw-myciel4				7200	7200	7200	7200	1468	1507	1460	1457
uc-case11				7200	7200	7200	7200	8966	8957	8946	8939
uc-case3				7200	7200	7200	7200	8397	8038	8018	7946
uct-subprob			✓	1065	1064	1066	1062	1479	1494	1481	1478
umts		✓		3966	2698	6103	3945	406	402	619	389
unitcal_7	✓		✓	275	274	274	274	4840	4820	4830	4850
usAbbrv-8-25_70				7200	7200	7200	7200	10781	10676	10683	10608
van				7200	7200	7200	7200	89471	93927	78965	70697
vpm1			✓	1	1	1	1	0	0	0	0
vpm2			✓	2	2	2	2	43	53	66	43
vpphard	✓			7200	7200	7200	7200	689845	690004	690144	689640
vpphard2				7200	7200	7200	7200	634470	634712	642961	634393
wachplan			✓	933	936	938	937	1680	1760	1830	1640
wnq-n100-mw99-14				7200	7200	7200	7200	531843	531683	531586	531843
zib01				30	31	30	30	3020	3097	3028	3025
zib02				7200	7200	7200	7200	0	0	0	0
zib54-UUE	✓	✓		2853	2291	2644	2310	1495	1545	1460	1502