# BIASED RANDOM-KEY GENETIC ALGORITHMS FOR THE WINNER DETERMINATION PROBLEM IN COMBINATORIAL AUCTIONS

C.E. ANDRADE, R.F. TOSO, M.G.C. RESENDE, AND F.K. MIYAZAWA

Abstract. In this paper, we address the problem of picking a subset of bids in a general combinatorial auction so as to maximize the overall profit using the first-price model. This winner determination problem assumes that a single bidding round is held to determine both the winners and prices to be paid. We introduce six variants of biased random-key genetic algorithms for this problem. Three of them use a novel initialization technique that makes use of solutions of intermediate linear programming relaxations of an exact mixed integer-linear programming model as initial chromosomes of the population. An experimental evaluation compares the effectiveness of the proposed algorithms with the standard mixed linear integer programming formulation, a specialized exact algorithm, and the best-performing heuristics proposed for this problem. The proposed algorithms are competitive and offer strong results, mainly for large-scale auctions.

## 1. Introduction

An auction is a mechanism or negotiation protocol for exchanging goods and services. In general, such goods are offered for bid, followed by a pre-determined round of bids, after which the highest bidder is pronounced the winner and pays for the negotiated item. Procurement auctions, on the other hand, are defined as follows: the auctioneer requests a set of goods, and each bidder can submit bids for this set. The lowest bidder is pronounced the winner and the auctioneer is paid for the goods. Today, auctions are widespread and, more importantly, distributed, thanks mainly to the Internet. Examples can be found in advertisement and position auctions in search engines such as Google and Yahoo!, as well as those coordinated by governments to negotiate radio spectrum, offshore oil and gas exploration, general goods, and services, among others.

In this paper, we are interested in general combinatorial auctions where bidders place bids (usually sealed) on subsets of goods, also known as *bundles*. Each bidder has access to a finite set of goods and is asked to come up with a list of bids, where each bid is an offer for a subset of goods. The objective of a bidder is to win the bid at an acceptable price. The greatest advantage of this type of auction is that it generates high economic efficiency since it allows the bidders to express both complementarity and substitutability of their preferences within bids. More formally, let $M$ be a set of goods, $g_1, g_2 \in M$ be two goods, and let

*Date*: August 2014.

*Key words and phrases.* Combinatorial auctions, winner determination problem, genetic algorithms, biased random-key genetic algorithms.

AT&T Labs Report Technical Report. To appear in *Evolutionary Computation*.

$f : 2^M \to \mathbb{R}$ be a valuation function for sets of these goods. Goods $g_1$ and $g_2$ are said to be *complementary* if and only if $f(\{g_1\}) + f(\{g_2\}) \leq f(\{g_1, g_2\})$, where $\{g_1, g_2\}$ denotes a bundle of goods $g_1$ and $g_2$. They are said to be *substitutes* if and only if $f(\{g_1\}) + f(\{g_2\}) \geq f(\{g_1, g_2\})$. For other variations, see Parsons et al. (2011). Since we allow bids for any subset of goods, there could be as many as $n(2^m - 1)$ bids, where $n$ is the number of bidders and $m$ is the number of goods. Hence, one of the key problems arise in auction mechanisms is to determine the winners of the auction, i.e. selecting pairwise disjoint bids to maximize the sum of the values of the selected bids. For other associated problems, see Cramton et al. (2006).

We focus on the *Winner Determination Problem* or WDP. In general, the WDP is equivalent to the weighted set packing problem, a well-known $\mathcal{NP}$-hard problem (Garey and Johnson, 1979). Notice that solving the WDP in auctions with no additional constraints and where only simple bids are allowed (i.e., bids for a single good) can be easily done in $O(nm)$-time. The seminal work on the WDP is credited to Rothkopf et al. (1998), who identified several special cases that can be solved in polynomial time. Such cases involve special bid structures like bid trees, geometrical regions, and cardinal restricted bids. However, these structures limit the expressiveness of the bids, potentially leading to an inefficient economy (Bichler et al., 2009).

Refined exact approaches to solve the WDP were proposed by Sandholm (2002; 2006) and Escudero et al. (2009), who also presented a polyhedral study applying cuts in an exact algorithm that scaled well in auctions with up to 300 bids. With regard to approximation algorithms, the general case cannot be approximated by a factor of $O(m^{1/2+\epsilon})$ of the optimal total value of the selected bids (unless $\mathcal{P} = \mathcal{NP}$), a bound inherited from the set packing problem (see Halldórsson (2000), who also describes an algorithm with $O(\ell/(\log \ell)^2)$-approximation that runs in $O(\max(\ell^c, m^2\ell^2))$-time, where $\ell$ is the number of bids and $c$ is a constant). An approximation algorithm with factor $O(\sqrt{m})$ is described in Lehmann et al. (2002) for the case in which each bidder has interest in only one particular bundle. For more approximation algorithms for special formulations, see Dobzinski et al. (2005) and Feige and Vondrák (2010). Several such algorithms and special cases of the winner determination problem are revisited in Blumrosen and Nisan (2007).

The first heuristic addressing the WDP specifically is the `Casanova` algorithm (Hoos and Boutilier, 2000), a multi-start stochastic local search algorithm that runs on top of greedy randomized initial solutions. A hill-climbing procedure can be found in Holte (2001). The first metaheuristic-based heuristics addressing the problem were a genetic algorithm and a simulated annealing heuristic (Schwind et al., 2003). A hybrid simulated annealing with local search called `SAGII` was proposed by Guo et al. (2006). To date, the strongest results come from a memetic algorithm by Boughaci et al. (2009) and Boughaci (2013).

It is interesting to observe that the WDP can also be modeled as the *Multidimensional Knapsack Problem* (MDKP), enabling the utilization of the algorithms developed to tackle the latter. As with the weighted set packing, the MDKP is a well-studied $\mathcal{NP}$-hard problem frequently used to evaluate new algorithms due to its intrinsic difficulty and its well-established benchmark test sets. One of the best heuristics to deal with MDKP was developed by Raidl and Gottlieb (2005) and consists in a genetic algorithm with weight-biased representation using surrogate duality to modify item weights. Recently, Mansini and Speranza (2012) presented an exact algorithm based on the idea of restricted core problems where a recursive variable fixing step is done until a given threshold is reached. The remaining subproblems are explored by a branch-and-bound approach. Several other approaches can be found, among them genetic algorithms (Chu and Beasley, 1998), tabu search (Vasquez and Vimont, 2005), ant-based optimization (Alaya et al., 2004), GRASP (Chardaire et al., 2001) and other hybridization techniques, e.g. Puchinger et al. (2010) and Boyer et al. (2010).

In this paper, we address the winner determination problem of general combinatorial auctions using the first-price model for single goods. We restrict ourselves to sealed auctions that use a single round to determine winners and prices to be paid, where the bids can be placed with no constraints other than their non-negativity. We also consider that the bids are anonymous and that bidder identity is not used to model or solve the underlying problem. Six variants of biased random-key genetic algorithms (BRKGAs) are implemented to address this problem, three of them adopting a novel scheme that employs linear programming (LP) relaxations to initialize the population when the underlying problem can be modeled as a 0–1 integer linear program. In such problems, an LP relaxation directly serves as a chromosome of the BRKGA heuristic, since both are defined over the interval $[0, 1]$. Experiments comparing the BRKGAs with a standard mixed integer-linear programming formulation of the WDP solved with a commercial solver, as well as the best performing heuristics proposed for the problem, the best performing exact algorithm, and the best performing heuristic for the MDKP, are carried out to identify the strengths and drawbacks of each approach.

The paper is organized as follows. In Section 2 we formalize the winner determination problem in combinatorial auctions. We then address biased random-key genetic algorithms in Section 3 and follow up with a description of our heuristics in Section 4. Experimental results are provided and discussed in Sections 6 and 7, respectively. Concluding remarks are made in Section 8.

## 2. General combinatorial auctions and their formulations

Several models for combinatorial auctions have been proposed in the literature, but most of them introduce additional constraints to limit the context of the auction so as to expose special properties that make the problem computationally easier. We next present a general description. Let $N = \{1, 2, \ldots, n\}$ be a set of bidders and $M = \{1, 2, \ldots, m\}$ be a set of goods. A collection of bids is represented by a tuple $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_n)$ such that $\mathcal{B}_i$ is the bid set of bidder $i$. Each bid $B \in \mathcal{B}_i$, for $i = 1, \ldots, n$, is a list of desired goods (bundle), i.e., $B \subseteq M$ such that bidder $i$ provides the function $b_i : 2^M \to \mathbb{R}^+$ that measures how much bidder $i$ is willing to pay for a bundle. An allocation of goods is represented by a tuple $\mathcal{S} = (\mathcal{S}_1, \ldots, \mathcal{S}_n)$

where $\mathcal{S}_i$ is the set of winner bids of bidder $i = 1, \ldots, n$. Note that

$$\left( \bigcup_{S \in \mathcal{S}_i} S \right) \cap \left( \bigcup_{R \in \mathcal{S}_j} R \right) = \emptyset, \text{ for all } i, j \in N.$$

We consider the *private information model* in which the auctioneer only knows the set of bids $\mathcal{B}$ and the functions $b_i$, for all $i = 1, 2, \ldots, n$. We restrict ourselves to first-price sealed auctions where the bidders submit one bid per desired bundle. This contrasts with *iterative auctions*, where the bidders may submit multiple bids to the same bundle in different rounds. Only the auctioneer can handle the bids and the winning bidders pay what they offered in their winning bids, i.e., $b_i(\mathcal{S}_i)$. Sealed auctions are used mainly in government and industry procurements. For more on the theory of auctions and its variants, see Krishna (2010).

The major work done in the literature has related the WDP with both the weight set packing problem and the multidimensional knapsack problem (see Bikhchandani and Ostroy (2010) for other models). In the set packing problem, we want to select weighted pairwise disjoint sets from a collection of items while maximizing the sum of the weights of the selected sets. A special case of this problem is the *Weighted Stable Set Problem* where we have a graph whose nodes have weights and one must choose a subset of nodes with no common incident edge and maximize the sum of weights. The WDP can be reduced to the weighted stable set problem in the following way: consider the intersection graph $G = (V, E)$, where each $s \in V$ represents a bid. An edge $(s, s') \in E$ exists if and only if $B_s \cap B_{s'} \neq \emptyset$, such that $B_s \in \mathcal{B}_i$, $B_{s'} \in \mathcal{B}_j$, $i, j \in N$ and $i \neq j$, i.e., the edge exists if and only if two bids from different bidders request a common good. A stable set on this intersection graph corresponds to a set of pairwise-disjoint bids from different bidders. Mathematically, the stable set problem can be written as the standard integer programming model

$$\begin{aligned} \max \quad & \sum_{s \in V} b_s x_s & & \\ \text{s.t.} \quad & x_s + x_{s'} \leq 1 & \forall (s, s') \in E & \qquad (1) \\ & x_s \in \{0, 1\} & \forall s \in V, & \end{aligned}$$

where we use the abbreviation $b_s = b(B_s)$, i.e., $b_s$ is the value offered for bundle $s$. Let the binary variable $x_s = 1$ if and only if bid $s$ is a winner. The above formulation enables overlapping among bids of a same bidder. Its main advantage is that the bidder in question need not present a bid for each subset of desired goods although the bidder may possibly overpay for some of them. In fact, this formulation is appropriate for super-additive valuations. The number of variables and constraints of this formulation is, respectively, $\ell$ and $\mathrm{O}(\ell^2)$, where $\ell$ is the total number of bids.

Another very common way to deal with WDP is to model it as a multidimensional knapsack problem. We consider that each bid is an item to be packed in the dimensions induced by the goods. Let $\hat{\mathcal{B}} = \bigcup_{i=1}^{n} \mathcal{B}_i$ be the set of all bids. In case two or more bids contain the same goods, we add a "dummy" good to each bid such that the new good uniquely identifies the bundle (Nisan, 2000). Let $w_{jk} = 1$

if good $j \in M$ is considered in bid $k \in \hat{\mathcal{B}}$, $w_{jk} = 0$, otherwise. The MDKP can be model as

$$
\begin{aligned}
\max \quad & \sum_{k \in \hat{\mathcal{B}}} b_k x_k \\
\text{s.t.} \quad & \sum_{k \in \hat{\mathcal{B}}} w_{jk} x_k \leq c_j \quad \forall j \in M \\
& x_k \in \{0,1\} \qquad\quad \forall k \in \hat{\mathcal{B}}.
\end{aligned} \tag{2}
$$

Again, we abuse the notation of $b_k$ as the value offered for bundle $k$ and we consider as winning bids, all $k$ such that $x_k = 1$. For combinatorial auctions with single goods, we have that $c_j = 1$ for all $j \in M$. A first observation is that this formulation can deal with multi-unit auctions where we can have multiple copies of good $j$ (by allowing $c_j \in \mathbb{N}$ for all $j \in M$) and a bid can request a certain number of copies of the good (by allowing $w_{jk} \in \mathbb{N}$, for $j \in M$ and $k \in \hat{\mathcal{B}}$). The number of variables and constraints of this formulation are, respectively, $\ell$ and $\mathrm{O}(m)$, where $\ell$ is the number of bids and $m$ is the number of goods.

The choice between the stable set and MDKP models can be very tricky in the case of single-unit combinatorial auctions. There are two important aspects to analyze: the tightness of the formulations and their sizes. With respect to tightness, it is well-known that the MDKP model (again, for the single-unit case) generates tighter formulations than the stable set model, since the former contains more clique inequalities than the latter and, therefore, results in better linear programming relaxations (Padberg, 1973). In fact, it is known that the stable set model, even with clique inequalities, leads to poor relaxations (Carr and Lancia, 2014).

With respect to size, although the MDKP formulation has the size $\mathrm{O}(\ell m)$, it can be much larger than the stable set formulation. The problem does not lie in the formulation itself, but in the input that can potentially be exponential for the MDKP in the number of goods. To illustrate this, suppose that a bidder has the following bids: $B_1 = (\{1,2\}, \$10)$, $B_2 = (\{2,3,4\}, \$10)$, and $B_3 = (\{4,5\}, \$10)$. If we consider only these bids, the stable set formulation will have three variables and no constraint, since the overlapping bids belong to the same bidder. In the MDKP, the bidder must generate, besides the given bids, the bids $B'_{12} = (\{1,2,3,4\}, \$20)$, $B'_{13} = (\{1,2,4,5\}, \$20)$, $B'_{23} = (\{2,3,4,5\}, \$20)$, and $B'_{123} = (\{1,2,3,4,5\}, \$30)$ since the bidder cannot win overlapping bids in this model. Although we have only one constraint, the number of variables (bids) is exponential with respect to the those in the stable set model. Note that if bidder identities are unknown, we must consider each bid individually and the MDKP becomes the best choice. In this paper, we do not consider bidder identities and therefore adopt the MDKP model.

## 3. Biased random-key genetic algorithms

To search for good solutions for the winner determination problem, we implemented a biased random-key genetic algorithm (BRKGA) (Gonçalves and Resende, 2011a). Our choice was mainly grounded on recent successes with classical hard combinatorial optimization problems such as routing (Andrade et al., 2013), packing (Gonçalves and Resende, 2011b), clustering (Andrade et al., 2014), and others. Although the BRKGA is relatively new, the random-key idea has been used for several authors since the seminal work of Bean (1994). Norman and Bean (1999)

used a random-key genetic algorithm to solve complex scheduling problems; Raidl (1999) applied this technique to solve Multiple Container Packing problem; and Rothlauf et al. (2002) studied several representations of network trees using simple and random key encoded solutions.

Two key features distinguish BRKGAs from traditional genetic algorithms (Goldberg, 1989):

(1) A standardized chromosome encoding that uses a vector with $t$ uniformly drawn random keys (*alleles*) over the interval $[0, 1]$ (Bean, 1994);
(2) A well-defined evolutionary process which uses parameterized uniform crossover (Spears and DeJong, 1991) for exploitation and substitutes the application of the mutation operator on existing chromosomes with newly introduced *mutants* – defined as $t$-long vectors of (uniformly drawn) random keys – for exploration.

Notice that no task depends on the optimization problem for which a solution is being sought. In fact, the only connection between this metaheuristic and the underlying problem occurs when a chromosome is *decoded*, that is, when a solution to the problem is constructed from a chromosome from which the objective function value or *fitness* can be extracted for the sake of comparing distinct chromosomes. Analogously, decoders indirectly map the chromosome space $[0, 1]^t$ into the set of feasible solutions to the optimization problem. The pair formed by a chromosome and its fitness is called an *individual*.

Algorithm 1 summarizes a typical BRKGA framework. Basically, we generate $p$ chromosomes as initial individuals using vectors with $t$ uniformly drawn random keys over the interval $[0, 1]$. In each iteration, a problem-specific *decoder* extracts the fitness of the chromosomes. To build a new population, we copy the $p_e$ best individuals (called the *elite set*), add $p_\mu$ random chromosomes (the *mutants*), and generate $p - p_e - p_\mu$ offspring by applying the crossover operator. Crossover is done between a random individual from the elite set and an individual from the remainder of the population: an offspring is generated by *mating*, where we take each allele from the elite parent with probability $\rho_e$ or from the other parent with probability $1 - \rho_e$. With $\rho_e = 0.5$, the standard uniform crossover occurs. With $\rho_e > 0.5$ by definition, exploitation happens at two levels: when parents are selected, because one is drawn from the elite set, and when offspring are conceived, because their alleles are inherited from the elite parent with greater probability. Exploration happens with the introduction of mutants at each generation, since they are vectors with $t$ uniformly drawn random keys. The usual mutation operators on individual genes are not employed by the BRKGA. Observe that the above scheme prevents infeasibility since, by definition, the resulting chromosomes – both offspring and mutants – are always vectors of random keys over $[0, 1]$.

A common approach to genetic algorithms is the island model (Whitley et al., 1998), where several populations are evolved independently and exchange their best individuals every given number of generations. This improves the variability of individuals, usually speeding up convergence, and reduces the risk that the algorithm will get stuck in local optima. Note that it is not necessary that this process be done in parallel in the sense of using several parallel machines or CPUs. It is straightforward to adapt the BRKGA framework: $\pi$ separate populations are created such that they are evolved simultaneously applying the evolutionary process in lines 3–8 of Algorithm 1 to each population. In this case, we will have $P_1, \ldots, P_\pi$

populations, $E_1, \ldots, E_\pi$ elite sets, and $Q_1, \ldots, Q_\pi$ "next generation" pools. The individual exchanges occur when a given threshold is reached, for instance, at every $\delta$ generations. For each population $P_i$, the $\eta$ best individuals are copied from other populations $P_{j \neq i}$ and replace the $\eta(\pi - 1)$ worst individuals in $P_i$.

In conclusion, the parameters that must be specified beforehand are the size of the chromosomes $t$, the size of the population $p$, the size of elite set $p_e$, the number of mutants $p_\mu$ introduced at each generation, and the inheritance probability $\rho_e$. If using parallel populations, we must set the number of populations $\pi$ and the generation threshold $\delta$ to exchange the $\eta$ best individuals. Advice for the parameter setup can be found in Gonçalves and Resende (2011a).

## 4. Decoding the winner determination problem

We now focus on BRKGA decoders for the winner determination problem. Recall that an instance of the WDP consists of finite sets of bidders $N = \{1, \ldots, n\}$, goods $M = \{1, \ldots, m\}$, and a collection of bids $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_n)$, where $\mathcal{B}_i$ is the bid set of bidder $i \in N$. As aforementioned, we have not addressed the bidder identities; instead, we consider the bids $\hat{\mathcal{B}}$, as defined in Section 2, but in some fixed order such that $\hat{\mathcal{B}} = (B_1, B_2, \ldots, B_t)$, where $t = |\hat{\mathcal{B}}|$. Each bid $B_j$ has value $b_j$. The decoders select a subset of bids that is maximal with respect to the sum of their values while respecting the pairwise-disjoint constraints among selected bids.

We develop three approaches for decoding a solution. These approaches are related in how they select and analyze the bids based on chromosome values and structural information of the problem. Define the size of each chromosome to be $t$. Our decoders associate each bid with an allele, i.e., the value of the $j$-th random key is associated with the $j$-th bid. The first step is to sort the bids in some particular order, generating a permutation of bids.

**Chromosomal approach::** The keys are sorted in non-increasing order of their values. Ties are broken by element indices;

**Algorithm 1:** BRKGA scheme.

```
Generate the initial population P;
while a stopping criteria is not reached do
    Decode each chromosome of P and extract their solutions and fitness;
    Sort the population P in non-increasing order of fitness. Consider the top
    p_e individuals as the elite group E;
    Copy E to the next generation Q, unaltered;
    Add p_μ randomly-generated new chromosomes (mutants) to Q;
    Generate p − p_e − p_μ chromosomes (offspring) by parameterized crossover,
    selecting a random parent from E and another from P \ E. Add them to
    Q;
    P ← Q;
return best individual found.
```

**Greedy approach::** We first choose the keys whose values are greater than or equal to a threshold $\tau$; then, these keys are sorted in non-increasing order of the cost/benefit of their respective bids, i.e., $b_j/|B_j|$. Notice that, in the greedy approach, the relative order of the bids is fixed for all possible chromosomes and, in fact, a permutation of the bids is not generated. Instead, we generate an ordered list containing a subset of the original bids. Ties are broken by element indices;

**Surrogate Duality approach::** Similar to the greedy approach but the cost/benefit is calculated differently. Let $\alpha$ be the dual solution vector of the relaxation of Formulation (2) when $x \in [0,1]^t$. Note that each $\alpha_i$ is tied to good $i$ and represents the "shadow price" of $i$. The cost/benefit of $B_j$ is $b_j/\sum_{i \in B_j} \alpha_i$. This surrogate duality approach was first proposed by Pirkul (1987). As in the above greedy approach, the dual vector $\alpha$ may be computed only once, and ordered lists can then be generated from it. Again, ties are broken by element indices;

Note that in the greedy and surrogate approaches, the parameter $\tau$ induces an implicit binary chromosome encoding and does not take advantage of the magnitude of the keys as does the chromosomal approach. The rationale behind the greedy approach is that the algorithm will take the most efficient bundles at first. This means that it will prefer the bids that most value the goods individually. The surrogate duality approach tries to capture the aggregate consumption levels of goods, meaning that bid efficiency is a measure of how much the bid impacts the entire system when it is chosen as the winner. In other words, if the marginal cost of the goods for a given bid is high, then this bid considers goods with high demand and it may not be worthwhile to choose it as winner if it were to offer a low value for these goods.

As an example, consider the chromosome in the Figure 1. In chromosomal approach, we simply sort the keys generating a permutation of bids, as shown by the indices of the vector in Figure 1a. In the greedy and surrogate dual approaches, we first filter the bids by their keys (using $\tau = 0.5$ in this example) and then sort the remaining bids in non-increasing order of their cost/benefit (as shown in the grey vector in Figure 1b).

One can note that in the chromosomal approach, the chromosome is used to generate *a permutation* of bids to be used in the subsequent processing. In the greedy and surrogate dual cases, the chromosome is used to generate a *subset* of bids whose size is controlled by parameter $\tau$. Note that if $\tau = 0$, then all bids are considered at once and, as the relative order of bids is fixed a priori due the cost/benefit relation, the decoder always returns the same solution. This way, $\tau > 0$ can be viewed as a *separation threshold*. Lines 3–7 of Algorithm 2 summarize these procedures.

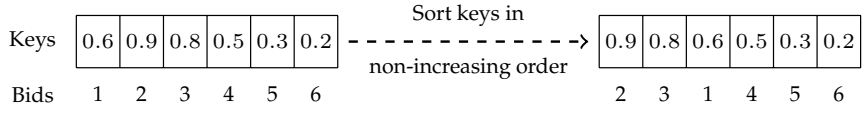
The next phase (lines 8–14) uses the sorted list of bids detailed above to construct a solution for the WDP. Initially, no bid is selected and all goods are unmarked. The decoder iterates over the bids in the supplied list, selecting a bid whenever all of its goods are not yet marked, thus maintaining the property that the winning bids are mutually exclusive with respect to their goods. If the current bid $B_j$ is selected, its corresponding goods are then marked. Otherwise, if bid $B_j$ has a conflicting good with another bid already selected, it is ignored and the value of the corresponding key, say $\kappa_j$, is reset to $1 - \kappa_j$ if $\kappa_j > 0.5$, discouraging this bid

from being considered in the following generations. Note that if $\kappa_j \leq 0.5$, this bid is already discouraged and its key value need not change.

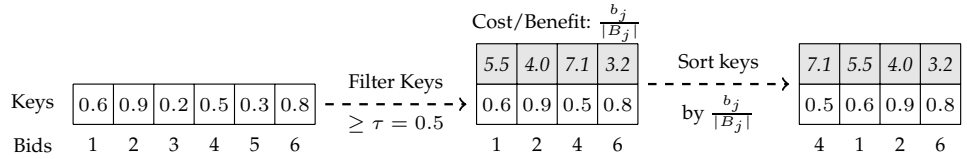
After this primary construction phase, the algorithm has checked all bids in the chromosomal approach. Therefore, in this case, it returns the solution value. In the greedy and surrogate dual cases, some bids are not visited because of filtering by $\tau$. Therefore, the algorithm builds a secondary list containing those bids that include only unmarked goods (disregarding those that were not selected in the previous phase due to conflicts). The bids are then sorted according to one of the above criteria, and the algorithm iterates over this list adding the bids that do not create conflict with the bids already selected. Each added bid $B_j$ has its corresponding key $\kappa_j$ reset to $1 - \kappa_j$ if $\kappa_j < 0.5$, encouraging this bid to be considered in further generations. This secondary phase is described in lines 17–24.

The running time for this procedure to obtain the sorted list of bids is bounded by $\mathrm{O}(t \log t)$, where $t$ is the number of bids.[1] In the worst case, the sum of the number of iterations in the two **foreach** loops is at most $t$, given that all bids may be analyzed. Checking and marking of goods can be implemented in $\mathrm{O}(1)$ using a simple binary vector indexed by the goods. This implies that, for each iteration, we have $\mathrm{O}(m)$ checks and markings in the worst case, where $m$ is the number of goods. Therefore, in the chromosomal case, we can bound the running time of the decoder by $\mathrm{O}(t \log t + tm)$.

The running times for the greedy and surrogate duality cases are different from the previous cases since we have an extra sorting procedure and a second traversal over the bids. As argued in the start of this section, the relative order of bids is fixed and need only be calculated once a priori. In this case, the sort procedures of lines 7 and 18 can be done in linear time using an indicator vector, where each position corresponds to the position of a bid in the pre-calculated order. Note that each sort procedure is done over a partition of the bids and, therefore, both together have running times that can be bounded by $\mathrm{O}(t)$. The first filtering traversal in




(A) Chromosomal Approach.




(B) Greedy Approach.

FIGURE 1. Example of sorting keys using the chromosomal and greedy approach.

[1]Note that this term depends on the sort algorithm used, and, in fact, can be reduced to $\Theta\left(t \frac{\log t}{\log \log t}\right)$ using fusion trees (Fredman and Willard, 1993).

**Algorithm 2:** Decoder for the Winner Determination Problem.

```
1  Let S be an empty list to hold the solution;
2  Let κ_j be the key associated with bid B_j;
3  if the chromosomal approach is used then
4  |  Let L be a list of bid indexes ordered in non-increasing order of keys κ;
5  else
6  |  Let L be a list of bid indexes such that κ_j ≥ τ for all bid B_j;
7  |  Sort L in non-increasing order of cost/benefit according greedy or
   |  surrogate dual approach;
8  foreach j ∈ L in the given order do
9  |  if B_j has no marked goods then
10 |  |  S ← S ∪ {j};
11 |  |  Mark all goods of B_j;
12 |  else if κ_j > 0.5 then
13 |  |  κ_j ← 1 − κ_j ; // discourage bid B_j
14 |  L ← L \ {j};
15 if the chromosomal approach is used then
16 |  Go to Line 25;
   // Process the remaining bids
17 Let L' be the list of indexes of remaining bids with unmarked goods;
18 Sort L' in non-increasing order of cost/benefit according greedy or surrogate
   dual approach;
19 foreach j ∈ L' do
20 |  if B_j has no marked goods then
21 |  |  S ← S ∪ {j};
22 |  |  Mark all goods of B_j;
23 |  |  if κ_j < 0.5 then
24 |  |  |  κ_j ← 1 − κ_j ; // encourage bid B_j
25 return the fitness Σ_{j∈S} b_j.
```

line 6 takes $t$ steps. The second traversal in line 17 is a function of $\tau$ and takes less than $t$ steps. Both loops together take $t$ iterations over $m$ goods. Thus, we can bound the total running time of these decoders by $\mathrm{O}(t) + 2t + tm = \mathrm{O}(tm)$.

## 5. Initializing the population of BRKGA

The most common approach to initialize the population of a BRKGA is to generate its chromosomes with uniformly drawn random keys over the interval $[0, 1]$. This results in highly heterogeneous individuals that may slow down the convergence of the algorithm.

In an attempt to speed up the search, we introduce a novel approach where we use solutions to the linear programming (LP) relaxations of Equation (2) as chromosomes, given that the decision variables of these relaxations are such that $0 \leq x_k \leq 1$ for all $k \in \hat{\mathcal{B}}$, where $\hat{\mathcal{B}}$ represents the set of all bids and $t = |\hat{\mathcal{B}}|$.

Therefore, a solution to the relaxed LP is a vector $x \in [0,1]^t$ that is compatible with the requirement of the keys of a BRKGA, and therefore we simply use the values of the optimal relaxed variables $x_k$ as the corresponding alleles of an initial chromosome. An advantage of using such an individual in the initial population is that it is perhaps closer to a good solution than are most random individuals. In addition to the pure relaxation, we use relaxations generated by the insertion of cutting planes in the original formulation. A *cutting plane* is an inequality that eliminates an infeasible solution for the original integer program. The insertion of cutting planes leads to tighter formulations with respect to the integer solutions (see e.g. Wolsey (1998) for more details). We expect that chromosomes generated from these tighter relaxations will be decoded into solutions that are even closer to good integer solutions.

This process consists in two nested phases as shown in Algorithm 3. In the first phase (lines 5–7), cutting planes are generated and added to the formulation and its linear relaxation solved. This results in vector $\tilde{x}$ such that $0 \leq \tilde{x_k} \leq 1$, for all $k = 1, \ldots, t$ except the fixed variables which have their values defined in next phase. Cut-generation procedures have been widely studied in the mathematical programming literature and can be implemented in different ways. In this paper, we do not make use of any particular cut-generation procedure but, rather, delegate their generation to the mixed integer programming (MIP) solver. To date, most modern MIP solvers, such as IBM ILOG Cplex, Gurobi Optimizer, and Fico Xpress are able to generate general strong cuts, such as clique cuts (Nemhauser and Wolsey, 1988) and Gomory fractional cuts (Gomory, 1958) known for the tight relaxations they produce. The task of finding cutting planes and reoptimization can be time consuming and therefore we limit this procedure to at most a predetermined number of steps or stop after a maximum time limit is reached. The relaxed solution $\tilde{x}$ is added to the initial population.

To generate several chromosomes, we fix variables iteratively, generating other relaxations (lines 10–16). In alternating iterations, we fix some variable $x_s$ to 0, meaning that the corresponding bid will not belong to any solution, or to 1, implying that the corresponding bid will belong to all solutions. This way, two consecutively generated chromosomes enforce the decision to select or not select the bid in question. One can note that we fix the variables in the order that they appear in set of bids. Another possible strategy is to choose a variable to fix uniformly at random saving the last fixed variable to restore its bounds. Both types of variable fixing procedures do not guarantee any solution quality, but diversify the search. Note that in the first iteration of Algorithm 3, no variable is fixed and a full relaxation of the model is solved. It is also possible, although unlikely, that two or more distinct variable fixings result in the same relaxation. In this case, we discard the duplicates.

Although the initialization with LP relaxations can speed up the convergence of the BRKGA, the time to generate these initial chromosomes is not negligible. It is worthwhile mentioning that solving an LP relaxation is a polynomial-time process, but finding cutting planes can be slow in certain situations, and several practical issues can contribute to this slowdown (Wolsey, 1998). In this regard, we set the stopping criterion for this type of initialization to a specific running time or number of chromosomes, whichever comes first (line 4). The remaining chromosomes are generated at random as is usual in the standard BRKGA.

**Algorithm 3:** Initialization by LP relaxations.

```
Let x_1, ..., x_t be a vector such that x_k is the variable associated to bid B_k ∈ B̂;
Let P be the empty initial population;
k ← 1;  bound ← 0;
while k < t and a stopping criterion is not reached do
    while maximum cutting iterations or the time limit are not reached do
        Insert cutting planes in the formulation if possible;
        Solve the LP relaxation;
    Let x̃ ∈ [0,1]^t be the relaxed optimal solution;
    P ← P ∪ {x̃};
    // Do variable fixing
    Fix x_k to bound;
    if bound = 0 then
        bound ← 1;
        if k ≥ 2 then
            Unfix x_{k-1};
    else
        bound ← 0;  k++;
if P is not complete then
    Generate random chromosomes to complete P;
```

## 6. Experimental Setup

We conducted several experiments with three objectives. The first objective was to investigate the effectiveness of biased random-key genetic algorithms to find optimal solutions for instances where exact algorithms succeeded in finding one. The second was to evaluate the solution quality for those instances where an optimal solution could not be found. Finally, the third objective was to investigate the effectiveness of the initialization of BRKGA with LP relaxations. Throughout the experiments, we compare our results with state-of-the-art algorithms for the WDP and the MKDP.

6.1. **Instances.** For the following experiments, we use two sets of instances. We first generated several instances using the Combinatorial Auction Test Suite, or CATS (Leyton-Brown et al., 2011), a standard generator of instances for combinatorial auction, largely adopted in the literature. The advantage of this suite lies in its ability to generate instances for several scenarios, such as time-scheduling auctions, matching auctions, region-border auctions, and even legacy distributions used in earlier papers. We generated two blocks of instances: one of smaller instances containing from 40 to 400 bids whose number of goods vary between 10 and 100; and another comprised of larger instances with 1000 to 4000 bids and 256 to 1500 goods. Preliminary experiments showed that the number of goods does not considerably affect the running time of the algorithms. This fact was also

observed by Buer and Pankratz (2010). Henceforth, we set the number of goods to be smaller than the number of bids in the test problems, seeking auctions with relevant conflicts among the bids, i.e. with several bids competing for the same sets of goods.

From CATS, we used legacy distributions L2, L3, L4, L6, L7, and the "arbitrary," "matching," "paths," "regions," and "scheduling" distributions (a total of 10 classes). We did not use the L1 and L5 distributions due to problems generating non-dominated bids. These instances broadly cover general combinatorial auctions. For further details, see Chapters 18 and 19 of Cramton et al. (2006). For each distribution, we generated three instances of each type according to Table 1 using the default parameters supplied by CATS. We also used the CATS hard mode (`-default_hard` flag), that generates three instances with approximately 1024 bids and 256 goods for each distribution with the objective of being hard to solve. The suite does not generate hard instances for "path" distributions. In summary, we used CATS to generate 120 small and 117 large instances.

A drawback of CATS is that instances appear to be easy in the sense that they can generally be solved by exact algorithms in reasonable time (see Boughaci et al. (2009); Guo et al. (2006)). In fact, such studies adopted a set of instances provided by Lau and Goh (2002), which are indeed harder than instances generated by CATS. These instances were generated using several factors observed in real brokering systems as pricing of a bundle, preference of each bidder, and fairness of good distributions. We selected three classes of such instances and called them LG. Each class contains 100 instances, all having more than 1000 bids.

In short, we experimentally analyzed the proposed algorithms on 537 instances where 417 of them are large with respect to number of bids, i.e., they have more than 1000 bids. Table 1 summarizes the instances adopted in the subsequent analysis. The last line of this table shows the number of instances in each class.[2]

6.2. **Algorithms.** In our evaluation, we considered specialized algorithms for both the winner determination problem and the multi-dimensional knapsack problem. We test two exact algorithms and two heuristics, both considered to be state-of-the-art for both problems.

To tune the parameters of the heuristics, we use the *iterated racing* procedure (Birattari et al., 2010). This method consists in sampling configurations from a particular distribution, evaluating them using either the Friedman test or the $t$-test, and refining the sampling distribution with repeated applications of F-Race. We use the `irace` package (López-Ibáñez et al., 2011), implemented in R, for parameter tuning. For each heuristic, we use a budget of 2,000 experiments in the tuning procedure, where each experiment was limited to one hour. For this propose, we

TABLE 1. Instance classes and their sizes.

| | CATS | | | | | | | | LG | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Bids | 40 | 80 | 200 | 400 | 1000 | 1024† | 2000 | 4000 | 1000 | 1000 | 1500 |
| Goods | 10 | 10 | 50 | 50 | 256 | 256 | 512 | 1024 | 500 | 1000 | 1500 |
| # of insts. | 30 | 30 | 30 | 30 | 30 | 27 | 30 | 30 | 100 | 100 | 100 |

† Generated using `default_hard` flag.

[2]These instances can be found in `http://www.loco.ic.unicamp.br/instances/wdp.html`

chose one instance of each size from each CATS class, and ten instances from each LG class, totalling 109 instances.

*Boughaci et al. Memetic Algorithm* — $\texttt{BO}_{\texttt{MA}}$. Boughaci et al. (2009) present a specialized memetic algorithm for WDP. It is a genetic algorithm that uses random-key encoding tied to a local search procedure for exploitation. Their representation and decoding phase is similar to our chromosomal approach. But in the reproduction phase, the individuals are chosen to crossover only if their "differ" sufficiently based on a similarity metric. In this case, the similarity is the size of the intersection of the winning bid sets induced by these individuals.

For crossover, the algorithm chooses an individual $X$ from the set $C_1$ of best individuals and another individual $Y$ from a set $C_2$ that contains individuals with small similarity with respect to individuals in $C_1$. The crossover is done traversing the concatenation $XY$ and choosing no-conflict bids in this order. The local search that characterizes the memetic flavor is the following: With probability $wp$ choose the best bid (one that maximizes the auctioneer's revenue) or, with probability $1 - wp$, a random bid. This bid is added to the solution and all other conflicting bids are removed. This process is repeated for a given number of iterations and returns the best individual found.

This algorithm outperformed other algorithms for the WDP that were previously proposed in the literature (`Casanova` of Hoos and Boutilier (2000) and `SAGII` of Guo et al. (2006)) and, indeed, presents competitive results, as we show in next sections.

We use the original C implementation provided to us by the author of Boughaci (2013). A slight modification was done to their implementation to support timing limits. In parameter tuning, we use the following ranges: population size $pop\_size \in [300, 2000]$; $|C_1| \in [5, 20]$; $|C_2| \in [7, 30]$; $wp \in [0.1, 0.5]$; and maximum local search iterations $max\_lsi \in [100, 500]$. The best setup indicated by `irace` was: $pop\_size = 1400$; $|C_1| = 12$; $|C_2| = 24$; $wp = 0.3$; and $max\_lsi = 150$.

*Raidl and Gottlieb Weight-Biased Genetic Algorithm* — $\texttt{RG}_{\texttt{RK}}$. Raidl and Gottlieb (2005) proposed a genetic algorithm for the MDKP where a solution is represented by a weight-biased real vector using surrogate dual information in the decoding phase. The authors used several probability distributions to generate the biased vectors. Their experiments show that following a log-normal distribution often works best. The weighted vector $w$ is generated such that $w_j = (1+\gamma)^{\mathcal{N}(0,1)}$ where $\mathcal{N}$ denotes a normally distributed random number with mean 0 and unit standard deviation and $\gamma > 0$ is a parameter that controls the intensity of biasing. Thus, the item $j$ is biased by a new price $p'_j = p_j w_j$.

The decoding phase uses the approach of Pirkul (1987) and is similar to our surrogate dual ordering. In this case, the pseudo utility of a item $j$ is $u_j = p'_j / \sum_{i=i}^{m} \alpha_i r_{ij}$, where $m$ is the number of dimensions, $\alpha_i$ is the dual value associated with dimension $i$ and $r_{ij}$ is the demand of item $j$ in dimension $i$.

The offspring generation is done by selecting two parents via binary tournaments, performing uniform crossover in their characteristic vectors, flipping each bit with probability $1/n$ (mutation probability), performing repair if a capacity constraint is violated, and always applying local improvement. If such a new candidate solution is different from all solutions in the current population, it replaces the worst of them

only if the new candidate has a better fitness than the worst solution (Puchinger et al., 2010). This algorithm, to date, is one of the best heuristics for the MDKP.

We use the Java code provided to us by the authors of Pfeiffer and Rothlauf (2007). In parameter tuning with `irace`, we use the following ranges: population size $pop\_size \in [300, 2000]$; tournament size $tourn\_size \in [10, 30]$; and $\gamma \in [0.01, 0.20]$. The best results were obtained with: $pop\_size = 500$; $tourn\_size = 20$; and $\gamma = 0.15$.

*Mansini and Speranza Exact Algorithm* — `CORAL`. Mansini and Speranza (2012) presented an exact algorithm for MDKP using the idea of *core items*. This algorithm divides the problem into subproblems with a limited number of variables. For each subproblem, a recursive variable fixing procedure is applied trying to fix as many variables as possible. The remaining unfixed variables represent the core items for which it is difficult to decide whether they belong to an optimal solution. For these items, a *restricted core* problem is built and solved with a branch-and-bound procedure. To speed up the branch-and-bound, several pruning conditions are introduced. This algorithm has a non-trivial implementation and we omit several details here which can be found in the original publication.

`CORAL` is considered to be a state-of-the-art exact algorithm for the MDKP. It works particularly well on instances with a large number of items. Its key feature is the ability to continually improve lower bounds using the optimal solutions from their restricted subproblems. However, in instances with a large number of constraints, `CORAL` has difficulty in finding good solutions, as observed in Mansini and Speranza (2012).

We use the original Java implementation provided to us by the authors. Slight modifications were done to their implementation to support timing limits. All parameters were set as in the original paper.

*Standard Mixed Integer Programming Solver* — `CPLEX`. We also used the IBM ILOG CPLEX Optimizer as a standard mixed integer programming solver to deal with Formulation (2) directly. CPLEX uses a branch-and-cut algorithm which is a deterministic enumerative procedure that explores a solution space using a bounding process in the solution values of the tree built during the search. Further detail can be found in Wolsey (1998). This type of algorithm has exponential running time in the worst case. The implementation of the IBM ILOG CPLEX Optimizer uses linear programming relaxations to bound the solution values in addition to using primal heuristics to produce integer solutions. According to its documentation, the IBM ILOG CPLEX Optimizer is fully deterministic with default parameters that are used in our experiments.

Note that we use the cut generation procedure in our LP initialization approaches. In that situation, however, we only use solutions from the root node and the first level of the branching tree. In fact, we do not use CPLEX branching mechanism there but only solve the LP and apply cut generation procedures. All variable fixing is controlled by our procedure as shown in Algorithm 3.

We use the IBM ILOG CPLEX Optimizer version 12.5.0.0. All default control parameters were used, except time limit, which was set to 3,600 wall-clock seconds, and number of threads, set to four. Using the default settings, CPLEX performs a preprocessing step to try to eliminate variables and constraints and calculate initial

bounds. Unfortunately, we cannot know what methods are used to perform this preprocessing since CPLEX is a closed-source commercial package.

It is important to note that these settings are used only in the case where CPLEX is run stand-alone. To create the initial chromosomes for our algorithms based on LP relaxations, we set up CPLEX differently as described next.

*Our approaches.* Our algorithms are described as follow:

- $\texttt{CA}_{\texttt{RA}}$**::** The proposed algorithm, defined in Section 4, using the chromosomal approach and random initialization;
- $\texttt{CA}_{\texttt{LP}}$**::** The proposed algorithm using the chromosomal approach but initialized with the optimal variables from the LP relaxations;
- $\texttt{GA}_{\texttt{RA}}$**::** The proposed algorithm using the greedy approach and random initialization;
- $\texttt{GA}_{\texttt{LP}}$**::** The proposed algorithm using the greedy approach and initialized with the optimal variables from the LP relaxations;
- $\texttt{SD}_{\texttt{RA}}$**::** The proposed algorithm using the surrogate duality approach and random initialization;
- $\texttt{SD}_{\texttt{LP}}$**::** The proposed algorithm using the surrogate duality approach and initialized with the optimal variables from the LP relaxations.

The proposed algorithms were written in C++ on top of the BRKGA API of Toso and Resende (2012), which implements all of the problem-independent components described in Section 3. Random numbers were generated by an implementation of the Mersenne-Twister (Matsumoto and Nishimura, 1998) and we used the standard sort algorithm of the C++ Standard Template Library. In particular, the used version implements the introsort algorithm whose worst case running time is $O(n \log n)$ (Musser, 1997). Our algorithms used four cores for simultaneous decoding (see Section 3).

To tune the BRKGA parameters with `irace`, we used the following ranges: elite percentage $\in [0.10, 0.30]$; percentage of mutants introduced at each generation $\in [0.05, 0.20]$; probability of inheriting each allele from elite parent $\rho_e \in [0.5, 0.8]$; number of independent populations $\pi \in [1, 3]$; exchange interval $\delta \in [50, 200]$; and number of elite individuals in an exchange $\eta \in [1, 2]$. The population size was set to $p = \min(10t, 2000)$, where $t$ is the number of bids. The main reason for this upper bound is to bound the running time of each generation and allow the BRKGA to evolve for several generations. We noted that populations with over 2,000 individuals had slow convergence because of the time needed to evaluate each generation. This is true mainly on large instances. The tuning results obtained with `irace` were very close to the values suggested by Gonçalves and Resende (2011a). The elite size was set to $p_e = \lceil 0.20p \rceil$, the number of mutants to $p_\mu = \lfloor 0.15p \rfloor$, and inheritance probability to $\rho_e = 0.70$. We evolved $\pi = 3$ populations simultaneously and once every $\delta = 100$ generations, each population exchanged its $\eta = 2$ best solutions with the other populations.

For the greedy and surrogate duality approaches, we set the filter threshold $\tau = 0.5$. In these cases, we expect that half of the bids are assigned to the first phase allocation and the other half to the second phase. As the algorithm evolves, good bids will have random key values greater than or equal to $\tau$ and the impact of the second phase will diminish since bad bids will have their goods marked in

the first phase. Note that $\tau$ has more impact on initial and mutants chromosomes than on others since on the latter the random keys evolved to a better solution.

For the approaches with LP-based initialization, we relaxed the integrality constraints of Formulation (2) to $0 \leq x_k \leq 1$. Since this procedure is time consuming, they were restricted as follows. The number of initial chromosomes was set to $lp_{init} = \lfloor 0.1p \rfloor$. Note that in Algorithm 3, this number may be small, since it is limited by the number of bids and duplications. If we obtain no duplicates, $lp_{init} \leq 2t+1$, where $t$ is the number of bids (the additive factor 1 is due the initial unrestricted relaxation). We allow two iterations of cut generation or five seconds to generate each chromosome. Both cut generation and the solution of the LP relaxation were done with the IBM ILOG CPLEX Optimizer version 12.5.0.0. We do not use other CPLEX features, such as variable fixing and branch-and-bound.

6.3. **Computational environment and algorithm settings.** The experiments were conducted on identical machines with quad-core Intel Xeon E5530 2.4 GHz CPUs and 32 GBytes of RAM running GNU/Linux. Running times reported are UNIX real wall-clock times in seconds, excluding the effort to read the instance. Each run was limited to 3,600 seconds for all algorithms. There are two reasons behind our choice of time limit. On the application side, procurement auctions are often managed in a short period of time to limit the response time of the bidders in order to achieve economic efficiency (for further details, see Chapters 2 and 23 in Cramton et al. (2006)). On the algorithmic side, the exact approaches produced a pattern of slow convergence characterized by minimal decrease in the optimality gap throughout the execution on all instances where an optimal solution was not found. This pattern can be clearly identified in the first hour of computation according to preliminary experiments. Notice in the results that follow that some running times are slightly over 3,600 seconds. This is because we wait for each algorithm to complete its current iteration before actually stopping it.

For the heuristics, we use an additional stopping criterion: 1,000 generations without improvement of the best solution found so far. In preliminary experiments, no instance presented an offset greater than 1,000 between two subsequent improvements. In fact, the average offset was about $126.20 \pm 138.00$, and the largest offset was 791 iterations. This way, we reduced the total computation time though we may have also reduced the long term effect of mutation in $\texttt{BO}_{\texttt{MA}}$ and $\texttt{RG}_{\texttt{RK}}$, and perhaps more seriously, the effect of the introduction of mutants in the BRKGAs.

To compile the C/C++ code, we use the GNU `g++` compiler version 4.8.1 and `libstdc++` version 6.0.18. To compile the Java code, we use the Oracle Java 64-Bit JDK runtime environment version 1.7.0_45. To allow for full memory utilization, we run the Java bytecodes with JVM parameter `-Xmx32g`.

## 7. Experimental Results and Discussion

This section presents our experimental results. For `CPLEX` and `CORAL`, we performed one run per instance since both are exact and deterministic algorithms. For the remaining algorithms, we performed 30 independent runs for each instance. One can note that this experimental setup is huge, and in fact it took more than 100 CPU days running over 32 identical machines. Each experiment was conducted individually on one machine to ensure the algorithm had exclusive use of all of the machine's resources. This way, we minimized external effects. All reported times

are wall-clock times where we reported only the optimization time excluding the effort to load the instance and log the run.

7.1. **Comparing revenue.** To compare the algorithms with respect to revenue, it is necessary to scale the results since each instance can have very different revenue values and even different orders of magnitude. For each instance $\mathcal{I}$, let $\chi_{\mathcal{I}}$ be the set of values of the solutions found for $\mathcal{I}$, and $D_{\mathcal{I}} = \max(\chi_{\mathcal{I}}) - \min(\chi_{\mathcal{I}})$. The scaling is done by the simple transformation

$$\chi'_{\mathcal{I}} = \begin{cases} (x - \min(\chi_{\mathcal{I}}))/D_{\mathcal{I}} & \forall x \in \chi_{\mathcal{I}} \text{ and } D_{\mathcal{I}} > 0, \\ 1 & \text{otherwise.} \end{cases}$$

where $\chi'_{\mathcal{I}}$ is the set of scaled values. Note that all values are scaled to the range $[0, 1]$.

Using this scaling process, Figure 2 shows the distribution of revenues for each algorithm. The box plots show the location of the first quartile, the revenue median and the third quartile. The whiskers extend to the most extreme revenue no more than 1.5 times the length of the box. The dots are the outliers.

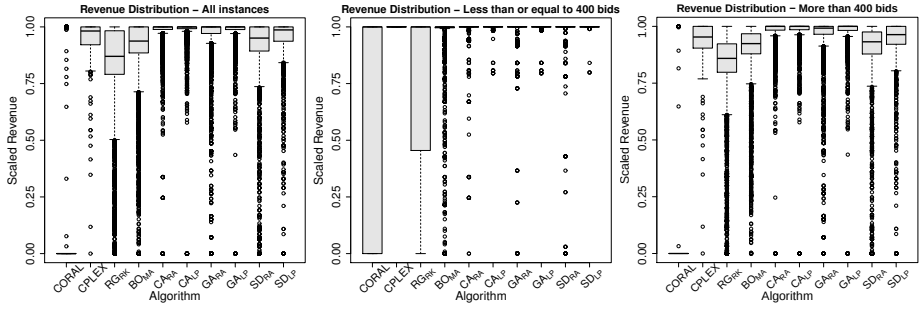
Figure 2a shows the revenue distribution over all instances. One can note that `CORAL` presented poor results when compared to the other algorithms. Indeed, this behavior was expected since Mansini and Speranza (2012) reported large solution times for instances with 500 bids (items, in their case). This can be clearly observed if we compare the distribution for small instances ($\leq$ 400 bids) in Figure 2b with the distribution for large instances ($\geq$ 1000 bids) in Figure 2c. `CORAL` is able to achieve a large range of values on small instances with a median of 0.99, close to the other results. For the large instances, `CORAL` rarely found a good solution, as shown by its outliers: it obtained only 20 optimal solutions on the large instances. One could argue that these instances are hard to solve by exact algorithms, however we observe that `CPLEX` does very well on them. `CPLEX`'s distributions are consistently good, with 113 optimal solutions found on large instances. The heuristics presented overall good results. Among them, $\texttt{RG}_{\texttt{RK}}$ experienced the worst results, followed by $\texttt{BO}_{\texttt{MA}}$. Our approaches did better than the other algorithms, although among our approaches, it is tricky to determine their relative performance simply examining the box plots. Note that the utilization of the pseudo utility derived from surrogate dual vectors did not, as expected, have a favorable impact on the results since $\texttt{SD}_{\texttt{RA}}$ and $\texttt{SD}_{\texttt{LP}}$ presented greater variances and lower medians than our other approaches that did not use surrogate duality.

The box plots help shape our intuition that our approaches obtained better results than those of previous methods. To confirm our conclusions, we tested the normality of these distributions using the Shapiro-Wilk test and applied the Mann-Whitney-Wilcoxon U test, considered more effective than the $t$-test for distributions sufficiently far from normal and for sufficiently large sample sizes (Conover, 1980; Fay and Proschan, 2010). For all tests, we assume a confidence interval of 99%. For small, large, and full distributions, the Shapiro-Wilk tests revealed that no revenue distribution fits a normal distribution since the $p$-values for all tests are less than $2.2 \times 10^{-16}$. Therefore, we applied the U test which assumes as null hypothesis that the location statistics are equal in both distributions. As several statistical tests were performed, we used a $p$-value correction procedure based on false discovery rate (FDR) to minimize the number of false positives (Type I error) as indicated by Benjamini and Hochberg (1995).

We tested the results of each pair of algorithms for small instances, large instances, and the full instance dataset.[3] For a confidence level of 99%, almost all comparisons were statistically significant, indicating differences among the results of the different algorithms (almost all $p$-values are less than 0.01). `CORAL` presented significantly poor performance and we believe there are two major reasons for this: first, `CORAL` cannot handle instances with a large number of bids and goods as previously shown by Mansini and Speranza (2012) and confirmed in Figure 2c. Second, `CORAL` takes advantage of the cardinality of each dimension constraint. In MDKP, each constraint $j$ is limited to be at most equal to $c_j$ (Formulation (2)), where, in general $c_j \geq 1$. In single-good WDPs, all constraints have cardinality $c_j = 1$, which does not allow `CORAL` to take advantage of them, limiting its major feature. Only on small instances does `CORAL` present good results, although worse than the other approaches.

`CPLEX` produced solid results that were, in general, better than those of $\texttt{RG}_{\texttt{RK}}$ and $\texttt{BO}_{\texttt{MA}}$. With respect to the algorithms proposed in this paper, `CPLEX` was worse except when compared to $\texttt{SD}_{\texttt{RA}}$. The comparison between `CPLEX` and $\texttt{SD}_{\texttt{LP}}$ was inconclusive ($p$-value $> 0.07$). For all small instances, `CPLEX` obtained an optimal solution. Although the U test returned a value of 0.00 for the differences among the algorithms and `CPLEX`, we believe that this difference is very small in favor of `CPLEX` since the other algorithms display more variance than does `CPLEX`, as shown in Figure 2b. Note that for the approaches using LP-based initialization applied to small instances, the tests against `CPLEX` were inconclusive ($p$-values $> 0.08$). We discuss this effect in Section 7.5. For large instances, its behavior was similar to that of the general case, where all instances are considered.

The performance of $\texttt{RG}_{\texttt{RK}}$ was worse than that of all heuristics and even `CPLEX`. This is surprising since this algorithm has been considered to be one of the best heuristics for the MDKP (Pfeiffer and Rothlauf, 2007). We argue that, like `CORAL`, $\texttt{RG}_{\texttt{RK}}$ uses in its favor the cardinality of each constraint since cardinality is correlated with the dual variables which are used to build the pseudo utilities. In the MDKP, this information is very rich since the multiplicity of the demand of each dimension is weighted by its corresponding dual cost. The WDP has unit cardinalities and therefore the pseudo utilities are less effective. Note that even in our BRKGAs, both




(A) (B) (C)

FIGURE 2. Dispersion of revenue for each algorithm.

[3]The full results are available in the supplementary material at `http://www.loco.ic.unicamp.br/results/wdp/supplementary_material.pdf`.

$\texttt{SD}_{\texttt{RA}}$ and $\texttt{SD}_{\texttt{LP}}$, which make use of surrogate duality, performed worse than other other variants which did not use this. $\texttt{BO}_{\texttt{MA}}$ performed quite well on the entire set of instances, producing results that are slightly worse than those of our approaches but better than previous algorithms.

Most variants of BRKGA outperformed all other algorithms in the general case. Among these variants, one can only point to small differences. The exception to this observation is $\texttt{SD}_{\texttt{RA}}$, whose location statistics are slightly lower those of CPLEX. It is surprising that the chromosomal approach (of $\texttt{CA}_{\texttt{RA}}$ and $\texttt{CA}_{\texttt{LP}}$), our most basic approach, showed better results than our other variants. We believe that this is so because the greedy and surrogate dual strategies can lead the algorithms to premature convergence to poor local maxima from which they are unable to escape. Note that for small instances, the tests comparing the LP-based initialized algorithms with CPLEX displayed $p$-values $> 0.08$ and, therefore, we cannot reject the hypothesis that these algorithms have similar performance. If we consider only the large instances, the behavior was similar to the general case. In general, the results for $\texttt{CA}_{\texttt{RA}}$ and $\texttt{GA}_{\texttt{LP}}$ are inconclusive since in their test the $p$-value $> 0.29$.

Table 2 reports the performance of the algorithms considering the instances partitioned into two sets. The first column is the name of the algorithm. The first group of columns (2–6) shows the performance considering 202 instances for which an optimal solution was found by CPLEX. There, column "# OPT" represents the number of instances for which the algorithm found an optimal solution; column "% Opt" shows a percentage of the number of optimal solutions found; and column "% Run" shows a percentage of the number of runs on which the algorithm found an optimal solution. The two columns under label "Prod. diff." show, respectively, the average of the proportional difference between the optimal solution value and the achieved value (%), and its corresponding standard deviation ($\sigma$). As we used the optimum solutions obtained by CPLEX, its entries are presented at the maximum levels (minimum levels, in columns 5–6). CORAL found few optimal solutions (25.74% of the 202 instances for which optimal solutions are known) while the other solutions it produced varied widely with respect to quality. Note that both "% Opt" and "% Run" have the same value since a single run was performed per instance (see Section 7, first paragraph). The heuristics performed well, finding

TABLE 2. Algorithm performance on instances with known and unknown optimum solutions.

| Alg. | Known Optima (202 instances) | | | | | Unknown Optima (335 instances) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Optima | | | Prop. diff. | | Best | | | Prop. diff. | |
| | # Opt | % Opt | % Run | % | $\sigma$ | # Best | % Best | % Run | % | $\sigma$ |
| CPLEX | 202 | 100.00 | 100.00 | 0.00 | 0.00 | 31 | 9.25 | 9.25 | 4.26 | 2.70 |
| CORAL | 52 | 25.74 | 25.74 | 24.49 | 35.41 | 3 | 0.90 | 0.90 | 58.36 | 16.44 |
| $\texttt{RG}_{\texttt{RK}}$ | 123 | 60.89 | 19.93 | 3.02 | 3.22 | 3 | 0.90 | 0.56 | 8.64 | 3.57 |
| $\texttt{BO}_{\texttt{MA}}$ | 128 | 63.37 | 18.90 | 2.96 | 4.00 | 93 | 27.76 | 4.05 | 4.92 | 2.76 |
| $\texttt{CA}_{\texttt{RA}}$ | 171 | 84.65 | 24.48 | 0.95 | 0.95 | 200 | 59.70 | 26.36 | 1.00 | 1.29 |
| $\texttt{CA}_{\texttt{LP}}$ | 187 | 92.57 | 30.33 | 0.79 | 0.80 | 201 | 60.00 | 24.51 | 1.02 | 1.28 |
| $\texttt{GA}_{\texttt{RA}}$ | 184 | 91.09 | 23.66 | 2.33 | 3.68 | 142 | 42.39 | 18.89 | 1.54 | 1.55 |
| $\texttt{GA}_{\texttt{LP}}$ | 186 | 92.08 | 28.87 | 0.89 | 0.84 | 200 | 59.70 | 23.67 | 1.14 | 1.36 |
| $\texttt{SD}_{\texttt{RA}}$ | 144 | 71.29 | 24.24 | 7.99 | 14.74 | 42 | 12.54 | 4.02 | 4.24 | 2.45 |
| $\texttt{SD}_{\texttt{LP}}$ | 186 | 92.08 | 37.17 | 1.29 | 1.09 | 64 | 19.10 | 5.90 | 3.52 | 2.23 |

more than 60% of the optimal solutions. With the exception of $\texttt{SD}_{\texttt{RA}}$, they were never off by more than 4% of the optimal. As expected, the approaches using LP-based initialization often found optimal solutions. However, in some cases they did not reach an optimal solution, suggesting that the relaxation and variable-fixing process not always produced chromosomes that when evolved are decoded into an optimal solution (see Section 7.5).

The second group of columns (7–11) of Table 2 reports the performance of the algorithms on the 335 instances where no optimal solution is known. It follows the same structure of columns 2–6 but instead of comparing the algorithms with the optimum solution values, we compared them using the best known solutions. CPLEX was able to find a best known solution on 9.25% of the 335 instances, while on the remaining instances it was about 4.26% off of the best values. CORAL and $\texttt{RG}_{\texttt{RK}}$ only found three best known solutions while the average gaps of the solutions it found with respect to the best known solution values were about 58.36% and 8.64%, respectively. Again, we emphasize that the pseudo utility approach did not work well since $\texttt{RG}_{\texttt{RK}}$, $\texttt{SD}_{\texttt{RA}}$, and $\texttt{SD}_{\texttt{LP}}$ presented the worst results among all heuristics. The LP-based initialization approaches again found the best results (with the exception of $\texttt{SD}_{\texttt{LP}}$), but not as good as the quality of the solutions it found on instances with known optima.

7.2. **Iterations and runtime analyses.** Table 3 shows the average number of iterations taken by the heuristics to find a best solution. The last iterations without improvement performed in the value of the best solution found are disregarded. The first two columns of this table list, respectively, the instance classes and their corresponding sizes. Each following pair of columns shows the average number of iterations to find a best solution and standard deviation for each algorithm, respectively. For instances with 40 and 80 bids, all algorithms converged very early to an optimal solution. An analysis of $\texttt{CA}_{\texttt{RA}}$, the "most random" of all algorithms, show that all instances having 40 and 80 bids are easy. This is so because all but six runs shows only a single iteration to reach an optimal solution on 40 bids instances (one took three iterations and another five took two). Note that for the LP-based approaches on instances with 40 and 80 bids, the BRKGA framework did not play any role in the optimization since all runs took a single iteration (see Section 7.5).

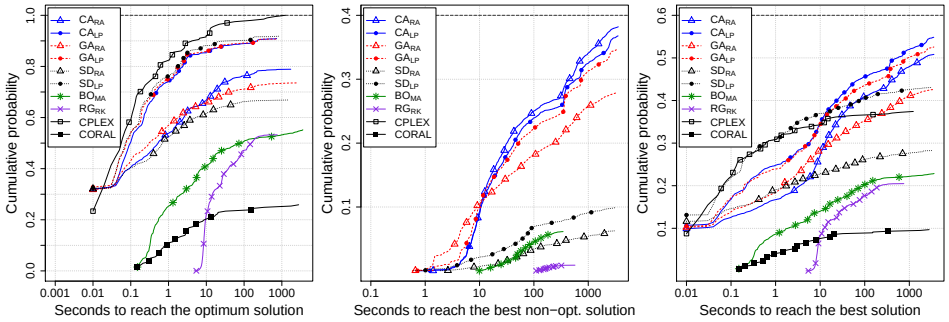
Figure 3 shows performance profiles (Dolan and Moré, 2002) for all algorithms. In performance profiles, the abscissa shows the time needed to reach a target solution value (in log scale), while the ordinate shows the cumulative probability to reach a target solution value for the given time in the abscissa. Each algorithm is characterized by a different performance profile curve made up of (time, cumulative probability) pairs, one for each execution of the algorithm on a particular instance. Runs that took over 3,600 seconds are not shown in the figure. Therefore, the percentage of runs that concluded within the time limit can be seen as the intersection of the profile with the right hand side of the figure.

Figure 3a shows performance profiles considering only target values of instances for which an optimum solution was found. Figure 3b has as target the values of the best solution found on instances with unknown optimal solution. Finally, Figure 3c takes, as target, the values of best solutions found for all instances. The solid black line with white squares shows the performance profile for CPLEX. As previously reported, CPLEX is quite fast to find these optimal solutions: in 82% of runs it required less than one second and in 99% less than 1,000 seconds. Only one run

TABLE 3. Average of iterations in finding the best solution. The last iterations without improvement in the best solution found are disregarded.

| Class | Size | $\texttt{RG}_{\texttt{RK}}$ | | $\texttt{BO}_{\texttt{MA}}$ | | $\texttt{CA}_{\texttt{RA}}$ | | $\texttt{CA}_{\texttt{LP}}$ | |
|---|---|---|---|---|---|---|---|---|---|
| | | Iter. | $\sigma$ | Iter. | $\sigma$ | Iter. | $\sigma$ | Iter. | $\sigma$ |
| CATS | 40 | 2 | 0.07 | 2 | 14.25 | 1 | 0.14 | 1 | 0.00 |
| | 80 | 2 | 0.47 | 21 | 95.38 | 1 | 0.45 | 1 | 0.00 |
| | 200 | 17 | 64.24 | 287 | 516.44 | 29 | 114.36 | 1 | 0.00 |
| | 400 | 64 | 167.19 | 348 | 675.30 | 109 | 227.47 | 30 | 137.06 |
| | 1000 | 243 | 360.07 | 523 | 629.76 | 322 | 456.50 | 118 | 271.27 |
| | 1024 | 257 | 345.40 | 480 | 561.63 | 284 | 448.99 | 121 | 282.51 |
| | 2000 | 232 | 204.81 | 435 | 572.54 | 651 | 767.01 | 220 | 422.79 |
| | 4000 | 121 | 86.07 | 155 | 147.78 | 579 | 521.75 | 319 | 459.41 |
| LG | 1000 | 95 | 195.78 | 509 | 515.63 | 197 | 311.99 | 197 | 317.58 |
| | 1500 | 46 | 91.45 | 469 | 531.82 | 178 | 258.24 | 166 | 240.78 |
| | | $\texttt{GA}_{\texttt{RA}}$ | | $\texttt{GA}_{\texttt{LP}}$ | | $\texttt{SD}_{\texttt{RA}}$ | | $\texttt{SD}_{\texttt{LP}}$ | |
| | | Iter. | $\sigma$ | Iter. | $\sigma$ | Iter. | $\sigma$ | Iter. | $\sigma$ |
| CATS | 40 | 1 | 0.55 | 1 | 0.00 | 1 | 2.65 | 1 | 0.00 |
| | 80 | 4 | 38.27 | 1 | 0.00 | 2 | 8.25 | 1 | 0.00 |
| | 200 | 44 | 142.41 | 1 | 0.00 | 76 | 182.31 | 1 | 0.00 |
| | 400 | 71 | 189.38 | 20 | 115.19 | 93 | 215.54 | 22 | 104.42 |
| | 1000 | 322 | 511.15 | 143 | 361.45 | 352 | 447.98 | 106 | 256.41 |
| | 1024 | 308 | 450.09 | 127 | 277.23 | 287 | 449.97 | 100 | 270.16 |
| | 2000 | 585 | 689.49 | 209 | 399.79 | 700 | 765.56 | 153 | 330.10 |
| | 4000 | 748 | 745.53 | 393 | 567.50 | 777 | 643.72 | 413 | 637.15 |
| LG | 1000 | 173 | 286.25 | 194 | 303.76 | 262 | 371.30 | 186 | 303.13 |
| | 1500 | 140 | 231.15 | 150 | 224.75 | 212 | 280.29 | 171 | 248.40 |

took 1,230 seconds. Since CPLEX spent about one hour on instances with unknown optima, it does not appear on Figure 3b. In general, CPLEX has the empirical probability of approximately 37% to find a best solution in less than 1,000 seconds. CORAL is represented by the solid black line with filled squares. It found 20% of optimal solutions in less than ten seconds but only 26% in less than 3,600 seconds. For the same reason as CPLEX, CORAL does not appear in Figure 3b and has around a 10% probability of finding a best solution in less than 1,000 seconds. $\texttt{BO}_{\texttt{MA}}$ (solid green line with asterisks) and $\texttt{RG}_{\texttt{RK}}$ (solid purple line with crosses) are slower than the other algorithms (except CORAL) in most cases. Note that $\texttt{BO}_{\texttt{MA}}$ has a small advantage over $\texttt{SD}_{\texttt{RA}}$ (dense dashed black line with triangles) when we consider Figure 3b. $\texttt{RG}_{\texttt{RK}}$ and $\texttt{BO}_{\texttt{MA}}$ found about 55% of the optimal solutions in less than 3,600 seconds. But, in general, $\texttt{BO}_{\texttt{MA}}$ presents a slightly better probability than that of $\texttt{RG}_{\texttt{RK}}$, as shown in Figure 3c. In general, BRKGA variants using LP-based initialization (lines with solid dots) are slower in the first ten seconds, due to the initialization process, but outperformed their corresponding counterparts after this. This fact is due to the time needed to create the first LP-based individuals. In fact, the average time of this procedure is $50.71 \pm 78.13$ seconds and the maximum time was 1377 seconds. The 377 additional seconds are due to instance setup as a CPLEX model. Considering optimal solutions, CPLEX presents the best time/probability tradeoff. Among the heuristics, $\texttt{SD}_{\texttt{LP}}$ (dense dashed line with solid dots) presents the highest probability (approximately 92%). Considering instances with unknown optima, $\texttt{CA}_{\texttt{RA}}$ (solid blue




(A) (B) (C)

FIGURE 3. Running time distributions to optimal assignment of winner bids. The identification marks correspond to 0.2% of the points plotted for each algorithm.

line with triangles) presents the highest probability (approximately 38%). Overall, the best empirical probability was approximately 55% for $\texttt{CA}_{\texttt{LP}}$ (solid blue line with solid dots).

7.3. **Comparing the heuristics on hard instances.** Since the exact methods can only solve to optimality the small or easy instances, the heuristics play a major role in solving the large instances. The following analysis uses the set of LG 1500/1500 instances which proved themselves to be the hardest instances considered in this paper. All algorithms, except $\texttt{BO}_{\texttt{MA}}$, reached the time limit on most runs and presented a relatively small number of iterations.

Figure 4 shows the distributions of revenues considering only the heuristics on the LG 1500/1500 instances. The values were scaled in the same fashion as was done in Section 7.1. We also performed the U test for each pair of algorithms at confidence level of 99%.[4] For the pairs of algorithms $(\texttt{CA}_{\texttt{RA}}, \texttt{CA}_{\texttt{LP}})$, $(\texttt{CA}_{\texttt{LP}}, \texttt{GA}_{\texttt{LP}})$, and $(\texttt{SD}_{\texttt{RA}}, \texttt{SD}_{\texttt{LP}})$, we cannot reject the hypothesis that the results of each pair are similar, since the $p$-values obtained from the U tests are greater than 0.01. For the other pairs, the differences presented in Figure 4 are statistically significant. Note that the revenues of $\texttt{BO}_{\texttt{MA}}$ are inferior to those of the BRKGAs, which confirms our previous suspicion that $\texttt{BO}_{\texttt{MA}}$ converges prematurely, as shown by its number of iterations in Table 3. As in the previous analyses, the algorithms using surrogate duality presented results below those of the other approaches, while $\texttt{CA}_{\texttt{RA}}$, $\texttt{CA}_{\texttt{LP}}$, and $\texttt{GA}_{\texttt{LP}}$ found the best values.

[4]The full results are available in the supplementary material at `http://www.loco.ic.unicamp.br/results/wdp/supplementary_material.pdf`.

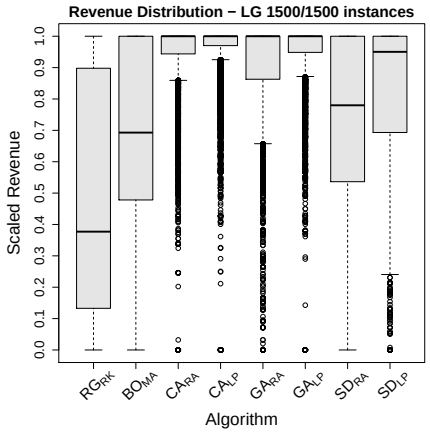
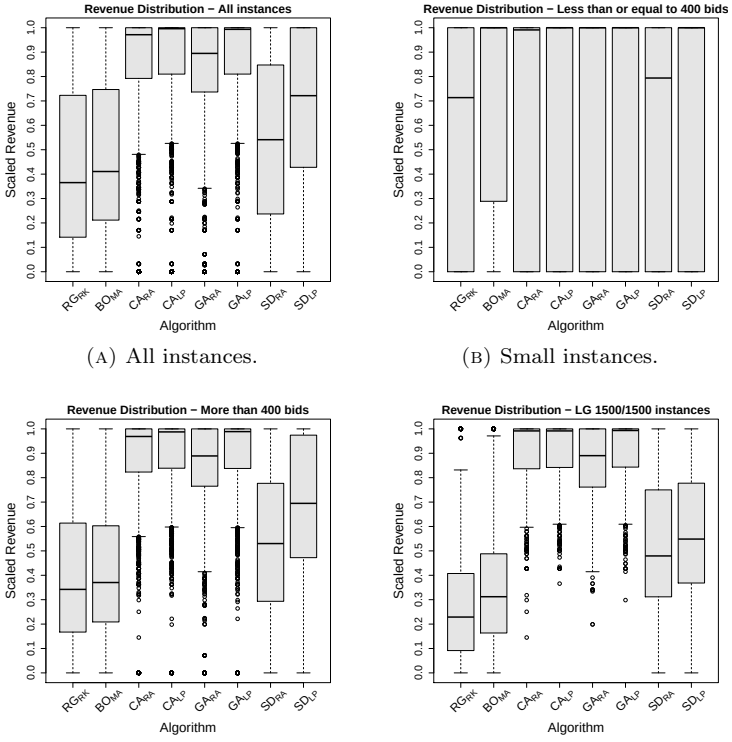



FIGURE 4. Dispersion of revenue for each algorithm on LG 1500/1500 instances.

7.4. **Comparing heuristics on small number of generations.** Standard genetic algorithms, such as $\texttt{RG}_{\texttt{RK}}$ and $\texttt{BO}_{\texttt{MA}}$, often converge quickly, i.e., in a small number of generations, to locally optimal, globally sub-optimal, solutions. Most of the diversification in their population is due to the initial population, since during evolution, new individuals are created only by crossover or mutation. Besides creating new individuals by crossover, BRKGAs insert new genetic material into the population at each generation in the form of mutants. This diversification can lead to long runs, i.e., having a large number of generations. Tables 2 and 3 show us that the BRKGAs and $\texttt{BO}_{\texttt{MA}}$ are able to find better solutions using, systematically, more iterations than $\texttt{RG}_{\texttt{RK}}$, suggesting that the adopted stopping criterion of 1,000 generations without improvement of the best solution may favor BRKGAs and $\texttt{BO}_{\texttt{MA}}$ over $\texttt{RG}_{\texttt{RK}}$.

To address this possible bias, we limit ourselves in this section to consider only the best solutions found by the algorithms in their first 100 generations. We chose 100 generations because this value is close to 95.49, the average number of generations taken by $\texttt{RG}_{\texttt{RK}}$ to first find the best solution in a given run. This way, we expect to reduce the impact of inserting new genetic material into the population of a BRKGA. We used the experimental results of previous sections but extracted the values after 100 generations. Note that for large instances, the algorithms were not able to reach the $100^{\text{th}}$ generation due the time limit. This is particularly true on large instances with 4,000 bids (and also on some instances with 2,000 bids). Therefore, these large instances are omitted from the following discussion.

Figure 5 shows the boxplots for these results. Their description is similar to Figure 2. We also performed U tests for each pair of algorithms using a confidence level of 99%.[4] In general, $\texttt{BO}_{\texttt{MA}}$ outperformed $\texttt{RG}_{\texttt{RK}}$ but not the BRKGAs, as observed in previous sections. The exception here is that $\texttt{BO}_{\texttt{MA}}$ is significantly better than all other algorithms on small instances (Figure 5b). In general, $\texttt{CA}_{\texttt{LP}}$ and $\texttt{GA}_{\texttt{LP}}$ presented the best results, although the test between them was inconclusive ($p$-value $> 0.84$). For large instances, $\texttt{CA}_{\texttt{LP}}$ and $\texttt{GA}_{\texttt{LP}}$ reached the best results. As before, we cannot affirm which of the two is best ($p$-value $> 0.77$). For hard LG 1500/1500 instances, the tests for $\texttt{CA}_{\texttt{RA}}$, $\texttt{CA}_{\texttt{LP}}$, and $\texttt{GA}_{\texttt{LP}}$ were inconclusive due $p$-values $> 0.66$.




(A) All instances.

(B) Small instances.

(C) Large instances.

(D) LG 1500/1500 instances.

FIGURE 5. Dispersion of revenue for each heuristic using 100 generations at most.

We conclude this section by observing that even with a small number of generations, the BRKGAs outperformed the other algorithms (except for $\texttt{BO}_{\texttt{MA}}$ on small instances). Another interesting observation was the performance of the algorithms with LP-initialization, which are able to produce better results than those with random initialization. We discuss this further in the next section.

7.5. **Effect of LP-based initialization.** One can notice in the tables of Section 7.1 and, with some difficulty in Figure 2, that the approaches using LP-based initialization performed better than the approaches that use only random vectors as the initial population. This could suggest that some chromosomes generated by the LP relaxation are often decoded into an optimum solution. In fact, this is not the case, as shown in Table 4. This table shows the average ratio $R_{LP}$ of the revenues of the best chromosome generated by LP relaxations and the best chromosome in the final population, i.e., $R_{LP} = best\_lp/best\_final$, for each algorithm and all the instances (hard and easy). The best LP-based chromosome values were obtained with Algorithm 3 and do not include any random individuals. Among all LP-based chromosomes, we select the one with the highest revenue. Note that we also consider random initialized algorithms. In this case, we consider the best randomly chromosome from the first generation.

TABLE 4. Ratio between the revenue of LP-based chromosomes and the best chromosome.

| Alg. | Size $\leq 400$ | | Size $\geq 1000$ | | All | |
|---|---|---|---|---|---|---|
| | Ratio | $\sigma$ | Ratio | $\sigma$ | Ratio | $\sigma$ |
| $\texttt{CA}_{\texttt{RA}}$ | 0.9848 | 0.02 | 0.9035 | 0.06 | 0.9183 | 0.06 |
| $\texttt{CA}_{\texttt{LP}}$ | 0.9999 | 0.00 | 0.9471 | 0.03 | 0.9574 | 0.03 |
| $\texttt{GA}_{\texttt{RA}}$ | 0.9963 | 0.00 | 0.9669 | 0.02 | 0.9724 | 0.02 |
| $\texttt{GA}_{\texttt{LP}}$ | 0.9999 | 0.00 | 0.9529 | 0.03 | 0.9617 | 0.03 |
| $\texttt{SD}_{\texttt{RA}}$ | 0.9931 | 0.01 | 0.9295 | 0.05 | 0.9427 | 0.05 |
| $\texttt{SD}_{\texttt{LP}}$ | 0.9998 | 0.00 | 0.9640 | 0.03 | 0.9721 | 0.03 |

Table 4 contains three blocks with respect to the size of the instances and each block has a column labeled "Ratio" that shows the average ratio $R_{LP}$ and a column labeled $\sigma$ with the corresponding standard deviation. One can note that on small instances, the LP-based chromosomes generate revenues very close to those in the final population, indicating a possible dominance of LP initial solutions. But note also that the ratio of random initialization and LP-based initialization revenues is very small. This ratio is about 4% larger when we consider large instances exclusively. It is interesting to note that $\texttt{GA}_{\texttt{RA}}$ displays a better ratio than that of $\texttt{GA}_{\texttt{LP}}$, implying that the results between the first and last generations of $\texttt{GA}_{\texttt{RA}}$ are closer together than the corresponding ones for $\texttt{GA}_{\texttt{LP}}$. However, $\texttt{GA}_{\texttt{LP}}$ presented better results than $\texttt{GA}_{\texttt{RA}}$ as shown in Section 7.1.

Table 5 presents the $R_{LP}$ ratios for the perspective instance class. Each column lists the average ratio of each instance class for each algorithm, with the exception of the last two, which are the overall average $r$ and standard deviation $\sigma$. The results are very close to those of Table 4. There are, however, some important points to consider. Note that on the instance class L2, Matching, and Scheduling, the LP-based initialization algorithms achieved a ratio of $R_{LP} = 1.00$ with a standard deviation of 0.00. This implies that for all instances of these classes, the best solution was found in the first generation. This, in turn, implies that the LP relaxations are able to produce the best solution. This is expected since such instance classes are known to be easy to solve. In fact, CPLEX was able to find the optimal solutions for these instances in the root node of the branch-and-bound tree, i.e. its heuristics were able to find these solutions without enumeration. In these cases, the evolutionary mechanism of BRKGA plays no role in the optimization. However, for other instance classes, BRKGA improves the solution value.

Note that for the CATS instances, the algorithms with LP-based initialization have very tight $R_{LP}$ ratios, i.e. around 0.99, especially when compared with the random initialization approach (for which they are around 0.95). For LG instances, the $R_{LP}$ ratios are larger than those for CATS: For both initialization approaches, the ratios are around 0.94. These results were expected since the CATS instances are easier than the LG instances. Again, it is interesting to note that $\texttt{GA}_{\texttt{RA}}$ presented tighter ratios than $\texttt{GA}_{\texttt{LP}}$ but only on the LG instances. This is not in contradiction with the results of Table 4 since the number of LG instances total more than half of all instances.

## 8. Concluding Remarks

In this paper, we introduced six variants of biased random-key genetic algorithms applied to the winner determination problem in combinatorial auctions. We also proposed a novel initialization scheme for BRKGAs based on intermediate solutions to the LP relaxation of the integer programming model for the that problem. Such scheme can be easily applied in BRKGAs for other 0–1 integer linear programs, since solutions for the LP relaxation of such problems are natural vectors of random-keys, and thus BRKGA chromosomes. The proposed algorithms have outperformed the standard LP model using a commercial mixed integer programming solver and other recent heuristic approaches on large CATS instances (Leyton-Brown et al., 2011), as well as on the LG instances (Lau and Goh, 2002). On small CATS instances, where optimal solutions were found with the commercial mixed integer programming solver but not the BRKGAs, the maximum gap between the two was less than 3%, showing that the BRKGAs can still obtain high-quality solutions. Another advantage of BRKGAs is their ability to find good solutions in a very short time, enabling their application in iterative auctions with thousands of goods and bids.

## Acknowledgments

The authors thank Dalila Boughaci for kindly providing the instances used in Lau and Goh (2002) and the source code of $\texttt{BO}_{\texttt{MA}}$. We also are grateful to Renata Mansini and Grazia Speranza for providing the source code of `CORAL`, and Jella Pfeiffer for provide her implementation of $\texttt{RG}_{\texttt{RK}}$. We thank the anonymous reviewers for providing comments that improved this paper significantly. Carlos E. Andrade is supported by São Paulo Research Foundation (FAPESP) grants 2010/05233-5 and 2012/08222-0. Flávio K. Miyazawa is supported by National Council for Scientific and Technological Development (CNPq) grants 306860/2010-4 and 477692/2012-5.

Table 5. Ratio between the revenue of LP-based chromosomes and the best chromosome by instance class.

| Alg | $\texttt{CA}_{\texttt{RA}}$ | $\texttt{CA}_{\texttt{LP}}$ | $\texttt{GA}_{\texttt{RA}}$ | $\texttt{GA}_{\texttt{LP}}$ | $\texttt{SD}_{\texttt{RA}}$ | $\texttt{SD}_{\texttt{LP}}$ | General | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | $r$ | $\sigma$ |
| L2 | 0.99 | **1.00** | 0.99 | **1.00** | 0.92 | **1.00** | 0.99 | 0.05 |
| L3 | 0.83 | 0.99 | 0.95 | 0.99 | 0.94 | 0.99 | 0.95 | 0.08 |
| L4 | 0.92 | 0.99 | 0.98 | 0.99 | 0.97 | 0.99 | 0.97 | 0.04 |
| L6 | 0.88 | 0.99 | 0.96 | 0.99 | 0.95 | 0.99 | 0.96 | 0.06 |
| L7 | 0.98 | 0.99 | 0.99 | 0.99 | 0.97 | 0.99 | 0.99 | 0.02 |
| Arbitrary | 0.89 | 0.96 | 0.95 | 0.96 | 0.93 | 0.97 | 0.94 | 0.05 |
| Matching | 0.89 | **1.00** | 0.97 | **1.00** | 0.96 | **1.00** | 0.97 | 0.06 |
| Paths | 0.90 | 0.99 | 0.96 | 0.99 | 0.95 | 0.99 | 0.96 | 0.05 |
| Regions | 0.89 | 0.97 | 0.95 | 0.97 | 0.93 | 0.97 | 0.94 | 0.06 |
| Scheduling | 0.99 | **1.00** | 0.99 | **1.00** | 0.99 | **1.00** | 0.99 | 0.01 |
| LG | 0.91 | 0.93 | 0.97 | 0.94 | 0.93 | 0.94 | 0.94 | 0.03 |

## References


Alaya, I., Solnon, C., and Ghédira, K. (2004). Ant algorithm for the multi-dimensional knapsack problem. In *International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2004)*, pages 63–72.

Andrade, C. E., Miyazawa, F. K., and Resende, M. G. C. (2013). Evolutionary algorithm for the $k$-Interconnected Multi-Depot Multi-Traveling Salesmen Problem. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, GECCO '13, pages 463–470, New York, NY, USA. ACM.

Andrade, C. E., Resende, M. G. C., Karloff, H. J., and Miyazawa, F. K. (2014). Evolutionary algorithms for overlapping correlation clustering. In *Proceeding of the sixteenth annual conference on Genetic and evolutionary computation conference*. To appear.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal On Computing*, 2(6):154–160.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300.

Bichler, M., Pikovsky, A., and Setzer, T. (2009). An Analysis of Design Problems in Combinatorial Procurement Auctions. *Business & Information Systems Engineering*, 1:111–117.

Bikhchandani, S. and Ostroy, J. M. (2010). *Combinatorial Auctions*, chapter From the Assigment Model to Combinatorial Auctions, pages 189–214. MIT Press.

Birattari, M., Yuan, Z., Balaprakash, P., and Stützle, T. (2010). F-race and iterated f-race: An overview. In *Experimental methods for the analysis of optimization algorithms*, pages 311–336. Springer Berlin Heidelberg.

Blumrosen, L. and Nisan, N. (2007). *Algorithmic Game Theory*, chapter Combinatorial auctions, pages 267–299. Cambridge University Press.

Boughaci, D. (2013). Metaheuristic approaches for the winner determination problem in combinatorial auction. In Yang, X.-S., editor, *Artificial Intelligence, Evolutionary Computing and Metaheuristics*, volume 427 of *Studies in Computational Intelligence*, pages 775–791. Springer Berlin Heidelberg.

Boughaci, D., Benhamou, B., and Drias, H. (2009). A memetic algorithm for the optimal winner determination problem. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 13:905–917.

Boyer, V., Baz, D. E., and Elkihel, M. (2010). Solution of multidimensional knapsack problems via cooperation of dynamic programming and branch and bound. *European Journal of Industrial Engineering*, 4:434–449.

Buer, T. and Pankratz, G. (2010). Solving a bi-objective winner determination problem in a transportation procurement auction. *Logistics Research*, 2:65–78.

Carr, R. D. and Lancia, G. (2014). Ramsey theory and integrality gap for the independent set problem. *Operations Research Letters*, 42(2):137–139.

Chardaire, P., McKeown, G. P., and Maki, J. A. (2001). Application of grasp to the multiconstraint knapsack problem. In Boers, E. J. W., editor, *Applications of Evolutionary Computing*, volume 2037 of *Lecture Notes in Computer Science*, pages 30–39. Springer Berlin Heidelberg.

Chu, P. C. and Beasley, J. E. (1998). A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4:63–86.

Conover, W. J. (1980). *Practical Nonparametric Statistics*. John Wiley & Sons, 2nd edition.

Cramton, P., Shoham, Y., and Steinberg, R. (2006). *Combinatorial Auctions*. MIT Press.

Dobzinski, S., Nisan, N., and Schapira, M. (2005). Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, STOC'05, pages 610–618, New York, NY, USA. ACM.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213.

Escudero, L. F., Landete, M., and Marín, A. (2009). A branch-and-cut algorithm for the winner determination problem. *Decision Support Systems*, 46(3):649–659.

Fay, M. P. and Proschan, M. A. (2010). Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics Surveys*, 4:1–39.

Feige, U. and Vondrák, J. (2010). The Submodular Welfare Problem with Demand Queries. *Theory of Computing*, 6(1):247–290.

Fredman, M. L. and Willard, D. E. (1993). Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences*, 47:424–436.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*. Freeman, San Francisco.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Professional, Boston, MA, USA, 1st edition.

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278.

Gonçalves, J. F. and Resende, M. G. C. (2011a). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17:487–525.

Gonçalves, J. F. and Resende, M. G. C. (2011b). A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem. *Journal of Combinatorial Optimization*, 22:180–201.

Guo, Y., Lim, A., Rodrigues, B., and Zhu, Y. (2006). Heuristics for a bidding problem. *Computers & Operations Research*, 33(8):2179–2188.

Halldórsson, M. M. (2000). Approximations of weighted independent set and hereditary subset problems. *Journal of Graph Algorithms and Applications*, 4(1):1–16.

Holte, R. (2001). Combinatorial Auctions, Knapsack Problems, and Hill-Climbing Search. In Stroulia, E. and Matwin, S., editors, *Advances in Artificial Intelligence*, volume 2056 of *Lecture Notes in Computer Science*, pages 57–66. Springer Berlin / Heidelberg.

Hoos, H. H. and Boutilier, C. (2000). Solving combinatorial auctions using stochastic local search. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 22–29. AAAI Press.

Krishna, V. (2010). *Auction Theory*. Academic Press, 2nd edition.

Lau, H. C. and Goh, Y. G. (2002). An intelligent brokering system to support multi-agent web-based 4th-party logistics. In *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence*, ICTAI '02, pages 154–, Washington, DC, USA. IEEE Computer Society.

Lehmann, D., O´callaghan, L. I., and Shoham, Y. (2002). Truth revelation in approximately efficient combinatorial auctions. *Journal of ACM*, 49:577–602.

Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., and Shoham, Y. (2011). CATS: The Combinatorial Auction Test Suite.

López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., and Birattari, M. (2011). The `irace` package, Iterated Race for Automatic Algorithm Configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium.

Mansini, R. and Speranza, M. G. (2012). CORAL: An exact algorithm for the multidimensional knapsack problem. *INFORMS Journal on Computing*, 24(3):399–415.

Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8:3–30.

Musser, D. R. (1997). Introspective sorting and selection algorithms. *Software: Practice and Experience*, 27(8):983–993.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and combinatorial optimization*. Wiley-Interscience, New York, NY, USA.

Nisan, N. (2000). Bidding and allocation in combinatorial auctions. In *Proceedings of the $2^{nd}$ ACM Conference on Electronic Commerce*, pages 1–12, New York, NY, USA. ACM.

Norman, B. A. and Bean, J. C. (1999). A genetic algorithm methodology for complex scheduling problems. *Naval Research Logistics (NRL)*, 46(2):199–211.

Padberg, M. W. (1973). On the Facial Structure of Set Packing Polyhedra. *Mathematical Programming*, 5:199–215.

Parsons, S., Rodriguez-Aguilar, J. A., and Klein, M. (2011). Auctions and bidding: A guide for computer scientists. *ACM Computing Surveys*, 43:10:1–10:59.

Pfeiffer, J. and Rothlauf, F. (2007). Analysis of greedy heuristics and weight-coded eas for multidimensional knapsack problems and multi-unit combinatorial auctions. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1529–1529, New York, NY, USA. ACM.

Pirkul, H. (1987). A heuristic solution procedure for the multiconstraint zero-one knapsack problem. *Naval Research Logistics (NRL)*, 34(2):161–172.

Puchinger, J., Raidl, G. R., and Pferschy, U. (2010). The multidimensional knapsack problem: Structure and algorithms. *INFORMS J. on Computing*, 22(2):250–265.

Raidl, G. R. (1999). The multiple container packing problem: A genetic algorithm approach with weighted codings. *SIGAPP Appl. Comput. Rev.*, 7(2):22–31.

Raidl, G. R. and Gottlieb, J. (2005). Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. *Evolutionary Computation*, 13(4):441–475.

Rothkopf, M. H., Pekec, A., and Harstad, R. M. (1998). Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147.

Rothlauf, F., Goldberg, D. E., and Heinzl, A. (2002). Network random keys-a tree representation scheme for genetic and evolutionary algorithms. *Evolutionary Computation*, 10:75–97.

Sandholm, T. (2002). Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1–2):1–54.

Sandholm, T. (2006). *Combinatorial Auctions*, chapter Optimal Winner Determination Algorithms, pages 331–361. MIT Press.

Schwind, M., Stockheim, T., and Rothlauf, F. (2003). Optimization heuristics for the combinatorial auction problem. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 3, pages 1588–1595.

Spears, W. M. and DeJong, K. A. (1991). On the virtues of parameterized uniform crossover. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236.

Toso, R. F. and Resende, M. G. C. (2012). A C++ application programming interface for biased random-key genetic algorithms. Technical report, AT&T Labs Research, 180 Park Avenue, Florham Park, NJ 07932 USA.

Vasquez, M. and Vimont, Y. (2005). Improved results on the 0-1 multidimensional knapsack problem. *European Journal of Operational Research*, 165(1):70–81.

Whitley, D., Rana, S., and Heckendorn, R. B. (1998). The island model genetic algorithm: On separability, population size and convergence. *J. of Computing and Information Technology*, 7:33–47.

Wolsey, L. A. (1998). *Integer programming*. Wiley-Interscience, New York, NY, USA.

## Supplementary material

### Appendix A. Instance tightness

An important aspect of instances for the winner determination problem (WDP) is their tightness. This metric is used by Chu and Beasley (1998) to craft instances of the Multidimensional Knapsack Problem (MDKP), largely used in the literature as the main benchmark for this problem. The tightness of a constraint $j$ is defined as

$$t_j = \frac{c_j}{\sum_{k \in \hat{\mathcal{B}}} w_{jk}}, \tag{3}$$

where $c_j$ is the availability of resource $j$ and $w_{jk}$ is the amount of resource $j$ requested by $k$, as defined in Formulation (2) of the main text. Note that for the WDP, the tightness is

$$t_j = \frac{1}{|\{B : j \in B, B \in \hat{\mathcal{B}}\}|}, \quad \forall j \in M, \tag{4}$$

by definition, i.e., the tightness is defined as the inverse of the number of bids that request a certain good. Note that a low $t_j$ indicates that good $j$ is required by several bids, probably increasing the problem difficulty.

In the Chu and Beasley MDKP instances, every constraint of a given problem has the same tightness, which is either 0.25, 0.5, or 0.75. For the WDP instances, tightness varies for each constraint and depends heavily on the type and size of the problems. For the most classes, as the size increases, tightness decreases, notably for the L2, L7, and LG classes. By definition, for some classes tightness is almost constant as, e.g. L3 and matching. Table 6 shows the average tightness of each constraint for each class and problem size. Note that the hard "path" instances are not shown since the CATS suite does not generate hard instances for "path" distributions.

TABLE 6. Average of instances tightness.

| Class | Size | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 40 | 80 | 200 | 400 | 1000 | 1024 | 2000 | 4000 |
| L2 | 0.347 | 0.341 | 0.094 | 0.098 | 0.019 | 0.026 | 0.012 | 0.008 |
| L3 | 0.333 | 0.333 | 0.333 | 0.333 | 0.333 | 0.208 | 0.333 | 0.333 |
| L4 | 0.506 | 0.436 | 0.507 | 0.420 | 0.555 | 0.605 | 0.514 | 0.511 |
| L6 | 0.433 | 0.381 | 0.355 | 0.314 | 0.351 | 0.578 | 0.351 | 0.344 |
| L7 | 0.543 | 0.471 | 0.111 | 0.109 | 0.015 | 0.068 | 0.009 | 0.004 |
| Arbitrary | 0.192 | 0.207 | 0.132 | 0.140 | 0.134 | 0.138 | 0.130 | 0.131 |
| Matching | 0.347 | 0.333 | 0.333 | 0.333 | 0.333 | 0.333 | 0.333 | 0.333 |
| Paths | 0.562 | 0.567 | 0.350 | 0.375 | 0.192 | — | 0.171 | 0.143 |
| Regions | 0.196 | 0.220 | 0.133 | 0.135 | 0.133 | 0.135 | 0.134 | 0.131 |
| Scheduling | 0.317 | 0.249 | 0.190 | 0.252 | 0.202 | 0.114 | 0.195 | 0.216 |
| Size | 1000/500 | | 1000/1000 | | 1500/1500 | | | |
| LG | 0.317 | | 0.249 | | 0.190 | | | |

## 2. Statistical tests

Tables 7–14, show U test results for each pair of algorithms and different instance sizes, at 99% of confidence level. The structure of these tables is as follows: Each row and column is indexed by one algorithm. Each element in the diagonal (bold) is the median of the corresponding algorithm. The upper-right diagonal elements are the differences in location statistics for each pair of algorithms. A positive difference indicates that the "row algorithm" has its location statistics higher (better) than the "column algorithm", and the negative difference is the opposite. The bottom-left diagonal elements are the $p$-values of each test. We omitted all $p < 0.01$ values, that indicate that the difference is statistically significant for those pairs. We also omitted confidence intervals since for all tests the values lie in these intervals and they are very narrow. For instance, in Table 7 we can see that the location statistics for `CPLEX` ($2^{\text{nd}}$ line) are higher (better) than for $\texttt{RG}_{\texttt{RK}}$ ($4^{\text{th}}$ column) since the value 0.0806 is positive. Since the $p$-value for this pair was omitted ($3^{\text{rd}}$ line, $3^{\text{rd}}$ column), the table indicates that `CPLEX` performed significantly better than $\texttt{RG}_{\texttt{RK}}$ in these tests. We chose to display a large number of significant digits since for some pairs of algorithms the differences are very small they are still statistically significant. This is the case, for example, of algorithms $\texttt{GA}_{\texttt{RA}}$ and $\texttt{SD}_{\texttt{LP}}$ in Table 7 where the difference is only 0.000009 but is still significant (in terms of the U test) in favor of $\texttt{GA}_{\texttt{RA}}$.

Since several tests were performed, we applied a $p$-value correction procedure based on false discovery rate (Benjamini and Hochberg, 1995) aiming to minimize the number of false positives (Type I error).

TABLE 7. Difference in median location for cost distributions for all instances, using a confidence interval of 99%. The omitted $p$-values are less than 0.0009.

| | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| CORAL | **0**.000000 | −0.950720 | −0.840038 | −0.916819 | −0.999908 | −0.999944 | −0.988866 | −0.999941 | −0.930584 | −0.961331 |
| CPLEX | | **0**.982822 | 0.080600 | 0.018096 | −0.001477 | −0.003086 | −0.000017 | −0.002008 | 0.004974 | −0.000078 |
| $RG_{RK}$ | | | **0**.875503 | −0.051401 | −0.114096 | −0.115350 | −0.104717 | −0.114523 | −0.062547 | −0.088295 |
| $BO_{MA}$ | | | | **0**.939643 | −0.051826 | −0.053154 | −0.042700 | −0.051845 | −0.000850 | −0.025455 |
| $CA_{RA}$ | | | | | **1**.000000 | −0.000036 | 0.000002 | 0.000014 | 0.034406 | 0.000695 |
| $CA_{LP}$ | | | | | | **1**.000000 | 0.000005 | 0.000037 | 0.035707 | 0.003093 |
| $GA_{RA}$ | | | | | | | **1**.000000 | −0.000067 | 0.026002 | 0.000012 |
| $GA_{LP}$ | | | | | $p > 0.29$ | | | **1**.000000 | 0.034429 | 0.001266 |
| $SD_{RA}$ | | | | | | | | | **0**.955629 | −0.010476 |
| $SD_{LP}$ | | $p > 0.07$ | | | | | | | | **0**.951900 |

TABLE 8. Difference in median location for cost distributions for instances with 400 bids or less, using a confidence interval of 99%. The omitted $p$-values are less than 0.004.

| | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| CORAL | **0**.999984 | −0.000062 | −0.000052 | −0.000033 | −0.000012 | −0.000042 | −0.000031 | −0.000037 | −0.000017 | −0.000039 |
| CPLEX | | **1**.000000 | 0.000083 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| $RG_{RK}$ | | | **1**.000000 | −0.000049 | −0.000057 | −0.000029 | −0.000070 | −0.000077 | −0.000007 | −0.000026 |
| $BO_{MA}$ | | | | **1**.000000 | −0.000039 | −0.000011 | −0.000041 | −0.000016 | −0.000046 | −0.000025 |
| $CA_{RA}$ | | | | | **1**.000000 | −0.000047 | 0.000012 | −0.000077 | 0.000063 | −0.000068 |
| $CA_{LP}$ | | $p > 0.09$ | | | | **1**.000000 | 0.000071 | 0.000022 | 0.000060 | −0.000089 |
| $GA_{RA}$ | | | | | $p > 0.25$ | | **1**.000000 | −0.000037 | 0.000041 | −0.000007 |
| $GA_{LP}$ | | $p > 0.08$ | | | | $p > 0.86$ | | **1**.000000 | 0.000083 | −0.000032 |
| $SD_{RA}$ | | | | | | | $p > 0.01$ | | **1**.000000 | −0.000076 |
| $SD_{LP}$ | | $p > 0.30$ | | | | | | | | **0**.999300 |

TABLE 9. Difference in median location for cost distributions for instances with 1000 bids or more, using a confidence interval of 99%. The omitted $p$-values are less than 0.001.

| | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| CORAL | **0**.000000 | −0.946779 | −0.856397 | −0.922194 | −0.999946 | −0.999927 | −0.990230 | −0.999915 | −0.931543 | −0.957204 |
| CPLEX | | **0**.953821 | 0.079737 | 0.022089 | −0.035372 | −0.035907 | −0.023595 | −0.034921 | 0.014744 | −0.000004 |
| $RG_{RK}$ | | | **0**.863907 | −0.056439 | −0.124063 | −0.124426 | −0.112816 | −0.123114 | −0.062318 | −0.088282 |
| $BO_{MA}$ | | | | **0**.926657 | −0.062474 | −0.062700 | −0.053631 | −0.061591 | −0.005172 | −0.029241 |
| $CA_{RA}$ | | | | | **1**.000000 | −0.000065 | 0.000065 | 0.000018 | 0.051891 | 0.027844 |
| $CA_{LP}$ | | | | | | **1**.000000 | 0.000086 | 0.000071 | 0.052948 | 0.028406 |
| $GA_{RA}$ | | | | | | | **0**.993222 | −0.000044 | 0.043039 | 0.017885 |
| $GA_{LP}$ | | | | | $p > 0.27$ | | | **1**.000000 | 0.051161 | 0.027183 |
| $SD_{RA}$ | | | | | | | | | **0**.938537 | −0.021078 |
| $SD_{LP}$ | | $p > 0.03$ | | | | | | | | **0**.951700 |

TABLE 10. Difference in median location for cost distributions for LG 1500/1500 instances, using a confidence interval of 99%. The omitted $p$-values are less than 0.00001.

| | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|
| $RG_{RK}$ | **0**.827570 | −0.081040 | −0.163875 | −0.162074 | −0.154572 | −0.160352 | −0.102957 | −0.109679 |
| $BO_{MA}$ | | **0**.910548 | −0.083920 | −0.083313 | −0.076227 | −0.079144 | −0.022304 | −0.028658 |
| $CA_{RA}$ | | | **1**.000000 | 0.000027 | 0.000036 | 0.000045 | 0.057928 | 0.053322 |
| $CA_{LP}$ | | | $p > 0.05$ | **1**.000000 | 0.000036 | 0.000057 | 0.057150 | 0.052600 |
| $GA_{RA}$ | | | | | **0**.998888 | −0.000042 | 0.050617 | 0.042824 |
| $GA_{LP}$ | | | | $p > 0.01$ | | **1**.000000 | 0.054981 | 0.047942 |
| $SD_{RA}$ | | | | | | | **0**.935822 | −0.005051 |
| $SD_{LP}$ | | | | | | | $p > 0.01$ | **0**.942900 |

TABLE 11. Difference in median location of cost distributions for all instances, considering the best solutions until 100 generations. A confidence interval of 99% was used. The omitted $p$-values are less than 0.000009.

| | $\texttt{RG}_{\texttt{RK}}$ | $\texttt{BO}_{\texttt{MA}}$ | $\texttt{CA}_{\texttt{RA}}$ | $\texttt{CA}_{\texttt{LP}}$ | $\texttt{GA}_{\texttt{RA}}$ | $\texttt{GA}_{\texttt{LP}}$ | $\texttt{SD}_{\texttt{RA}}$ | $\texttt{SD}_{\texttt{LP}}$ |
|---|---|---|---|---|---|---|---|---|
| $\texttt{RG}_{\texttt{RK}}$ | **0**.365256 | −0.027354 | −0.484062 | −0.495047 | −0.439106 | −0.495536 | −0.082448 | −0.251942 |
| $\texttt{BO}_{\texttt{MA}}$ | | **0**.410741 | −0.433579 | −0.443894 | −0.389594 | −0.444571 | −0.028155 | −0.208201 |
| $\texttt{CA}_{\texttt{RA}}$ | | | **0**.971370 | −0.000033 | 0.000045 | −0.000016 | 0.313978 | 0.115553 |
| $\texttt{CA}_{\texttt{LP}}$ | | | | **0**.995975 | 0.000040 | 0.000046 | 0.321063 | 0.124509 |
| $\texttt{GA}_{\texttt{RA}}$ | | | | | **0**.894743 | −0.000053 | 0.274147 | 0.069280 |
| $\texttt{GA}_{\texttt{LP}}$ | | | | $p > 0.84$ | | **0**.993376 | 0.321429 | 0.126418 |
| $\texttt{SD}_{\texttt{RA}}$ | | | | | | | **0**.541106 | −0.126061 |
| $\texttt{SD}_{\texttt{LP}}$ | | | | | | | | **0**.721500 |

TABLE 12. Difference in median location of cost distributions for instances with 400 bids or less, considering the best solutions until 100 generations. A confidence interval of 99% was used. The omitted $p$-values are less than 0.009.

| | $\texttt{RG}_{\texttt{RK}}$ | $\texttt{BO}_{\texttt{MA}}$ | $\texttt{CA}_{\texttt{RA}}$ | $\texttt{CA}_{\texttt{LP}}$ | $\texttt{GA}_{\texttt{RA}}$ | $\texttt{GA}_{\texttt{LP}}$ | $\texttt{SD}_{\texttt{RA}}$ | $\texttt{SD}_{\texttt{LP}}$ |
|---|---|---|---|---|---|---|---|---|
| $\texttt{RG}_{\texttt{RK}}$ | **0**.713545 | −0.000017 | −0.000011 | −0.000029 | −0.000019 | −0.000020 | −0.000009 | −0.000034 |
| $\texttt{BO}_{\texttt{MA}}$ | | **1**.000000 | 0.000005 | 0.000051 | 0.000030 | 0.000067 | 0.000038 | 0.000059 |
| $\texttt{CA}_{\texttt{RA}}$ | $p > 0.01$ | | **0**.990978 | −0.000027 | 0.000008 | −0.000006 | −0.000000 | −0.000015 |
| $\texttt{CA}_{\texttt{LP}}$ | | | $p > 0.01$ | **0**.999988 | 0.000003 | 0.000033 | 0.000021 | −0.000044 |
| $\texttt{GA}_{\texttt{RA}}$ | | | $p > 0.37$ | $p > 0.11$ | **0**.999943 | −0.000037 | 0.000018 | −0.000040 |
| $\texttt{GA}_{\texttt{LP}}$ | | | $p > 0.04$ | $p > 0.70$ | $p > 0.27$ | **0**.999986 | 0.000039 | −0.000049 |
| $\texttt{SD}_{\texttt{RA}}$ | $p > 0.41$ | | $p > 0.37$ | | $p > 0.09$ | $p > 0.01$ | **0**.793841 | −0.000020 |
| $\texttt{SD}_{\texttt{LP}}$ | | | | $p > 0.52$ | $p > 0.03$ | $p > 0.37$ | | **1**.000000 |

TABLE 13. Difference in median location of cost distributions for instances with more than 400 bids, considering the best solutions until 100 generations. A confidence interval of 99% was used. The omitted $p$-values are less than 0.0001.

| | $\texttt{RG}_{\texttt{RK}}$ | $\texttt{BO}_{\texttt{MA}}$ | $\texttt{CA}_{\texttt{RA}}$ | $\texttt{CA}_{\texttt{LP}}$ | $\texttt{GA}_{\texttt{RA}}$ | $\texttt{GA}_{\texttt{LP}}$ | $\texttt{SD}_{\texttt{RA}}$ | $\texttt{SD}_{\texttt{LP}}$ |
|---|---|---|---|---|---|---|---|---|
| $\texttt{RG}_{\texttt{RK}}$ | **0**.342195 | −0.020136 | −0.548485 | −0.557676 | −0.502515 | −0.556347 | −0.132994 | −0.300077 |
| $\texttt{BO}_{\texttt{MA}}$ | | **0**.370348 | −0.521230 | −0.531241 | −0.477689 | −0.530029 | −0.115826 | −0.284344 |
| $\texttt{CA}_{\texttt{RA}}$ | | | **0**.968853 | −0.000039 | 0.009928 | −0.000032 | 0.365621 | 0.198026 |
| $\texttt{CA}_{\texttt{LP}}$ | | | | **0**.987577 | 0.021097 | 0.000033 | 0.375408 | 0.208959 |
| $\texttt{GA}_{\texttt{RA}}$ | | | | | **0**.889071 | −0.020640 | 0.324260 | 0.154933 |
| $\texttt{GA}_{\texttt{LP}}$ | | | | $p > 0.70$ | | **0**.988826 | 0.375060 | 0.208850 |
| $\texttt{SD}_{\texttt{RA}}$ | | | | | | | **0**.530006 | −0.153962 |
| $\texttt{SD}_{\texttt{LP}}$ | | | | | | | | **0**.694700 |

TABLE 14. Difference in median location of cost distributions for for LG 1500/1500 instances, considering the best solutions until 100 generations. A confidence interval of 99% was used. The omitted $p$-values are less than 0.00004.

| | $\texttt{RG}_{\texttt{RK}}$ | $\texttt{BO}_{\texttt{MA}}$ | $\texttt{CA}_{\texttt{RA}}$ | $\texttt{CA}_{\texttt{LP}}$ | $\texttt{GA}_{\texttt{RA}}$ | $\texttt{GA}_{\texttt{LP}}$ | $\texttt{SD}_{\texttt{RA}}$ | $\texttt{SD}_{\texttt{LP}}$ |
|---|---|---|---|---|---|---|---|---|
| $\texttt{RG}_{\texttt{RK}}$ | **0**.228827 | −0.064896 | −0.661234 | −0.661403 | −0.608903 | −0.665038 | −0.242486 | −0.306383 |
| $\texttt{BO}_{\texttt{MA}}$ | | **0**.312164 | −0.599983 | −0.601567 | −0.544650 | −0.604113 | −0.172053 | −0.239376 |
| $\texttt{CA}_{\texttt{RA}}$ | | | **0**.991615 | −0.000039 | 0.009253 | −0.000065 | 0.418796 | 0.334902 |
| $\texttt{CA}_{\texttt{LP}}$ | | | $p > 0.97$ | **0**.991615 | 0.012762 | −0.000056 | 0.419685 | 0.334964 |
| $\texttt{GA}_{\texttt{RA}}$ | | | | | **0**.890265 | −0.018571 | 0.351878 | 0.290690 |
| $\texttt{GA}_{\texttt{LP}}$ | | | $p > 0.68$ | $p > 0.68$ | | **0**.993368 | 0.421907 | 0.336256 |
| $\texttt{SD}_{\texttt{RA}}$ | | | | | | | **0**.479321 | −0.057048 |
| $\texttt{SD}_{\texttt{LP}}$ | | | | | | | | **0**.548400 |

## 3. Additional running time results

Table 15 shows the average time in seconds taken by each algorithm to find the best solution (recall that we limited runs to at most 3,600 seconds). The additional time in the last iterations without improvement in the best solution found is disregard. We also exclude the time used loading instances and logging. To be fair with the Java implementations, each run began with a warm-up phase so that Java virtual machine could load and optimize all necessary bytecode. The first two columns of this table list, respectively, the instance classes and their corresponding sizes. Each following pair of columns shows the average time and standard deviation for each algorithm, respectively.

TABLE 15. Running time comparison among the algorithms. For each algorithm, it is shown the average time to find the best solutions. The over time used in the last iterations without improvement is disregard. Time in seconds.

| Class | Size | CORAL | | CPLEX | | $RG_{RK}$ | | $BO_{MA}$ | | $CA_{RA}$ | | $CA_{LP}$ | | $GA_{RA}$ | | $GA_{LP}$ | | $SD_{RA}$ | | $SD_{LP}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Time | $\sigma$ | Time | $\sigma$ | Time | $\sigma$ | Time | $\sigma$ | Time | $\sigma$ | Time | $\sigma$ | Time | $\sigma$ | Time | $\sigma$ | Time | $\sigma$ | Time | $\sigma$ |
| CATS | 40 | 1 | 1 | 1 | 1 | 10 | 6 | 1 | 1 | 1 | 0 | 1 | 0 | 2 | 1 | 1 | 0 | 2 | 3 | 1 | 0 |
| | 80 | 2 | 2 | 1 | 1 | 12 | 7 | 1 | 1 | 2 | 1 | 1 | 0 | 5 | 38 | 1 | 0 | 2 | 4 | 1 | 0 |
| | 200 | 792 | 1438 | 1 | 1 | 18 | 15 | 17 | 32 | 14 | 74 | 1 | 0 | 31 | 123 | 1 | 0 | 46 | 132 | 1 | 0 |
| | 400 | 923 | 1418 | 1 | 1 | 40 | 56 | 51 | 121 | 58 | 161 | 28 | 124 | 47 | 139 | 18 | 96 | 65 | 172 | 20 | 89 |
| | 1000 | 2883 | 1436 | 382 | 1075 | 473 | 651 | 736 | 936 | 53 | 136 | 30 | 96 | 60 | 134 | 36 | 104 | 75 | 167 | 32 | 112 |
| | 1024 | 2803 | 1494 | 960 | 1568 | 460 | 591 | 778 | 1130 | 54 | 142 | 33 | 114 | 72 | 165 | 30 | 106 | 79 | 181 | 28 | 96 |
| | 2000 | 2903 | 1411 | 1377 | 1708 | 1463 | 1151 | 1909 | 1324 | 72 | 181 | 35 | 106 | 66 | 135 | 39 | 114 | 78 | 166 | 29 | 99 |
| | 4000 | 3012 | 1344 | 1802 | 1799 | 2527 | 1540 | 2611 | 1361 | 52 | 115 | 28 | 75 | 68 | 133 | 42 | 119 | 54 | 123 | 30 | 87 |
| LG | 1000 | 3606 | 12 | 3601 | 1 | 289 | 529 | 75 | 68 | 94 | 196 | 93 | 192 | 94 | 196 | 95 | 197 | 78 | 181 | 71 | 168 |
| | 1500 | 3624 | 23 | 3601 | 1 | 425 | 702 | 66 | 58 | 118 | 211 | 113 | 200 | 94 | 187 | 100 | 187 | 92 | 188 | 83 | 179 |

## 4. Results for parameter tuning

**RGRK.**

```
# Best candidates
popsize tournamentsize pertubation
    434            18      0.1493
    472            16      0.1488
    437            16      0.1459
```

**BOMA.**

```
# Best candidates
popsize highqualityindividuas diversifiedindividuals maxlocalsearchiters
   1377                    12                      24                 142
    516                    16                      26                 238
   1178                    11                      24                 115
```

**BRKGAs.**

```
# Best candidates
     pe     pm   rhoe indpop intervalexchange elitexchange
0.2116 0.1639 0.7609      3              136            2
0.2527 0.0679 0.7698      3              127            1
0.2297 0.0741 0.7976      3              141            1
```

## 5. Best results for each instance

This section presents the results obtained for LG instances. The tables format is the following: the first column and second columns are the instance name and the best revenue obtained for this instance, respectively. The following columns show the percentage of the revenue from the best solution obtained by the algorithm that names the column. A high percentage indicates that the obtained solution is closer to the best. A star ($\star$) indicates that the algorithm found the best solution.

Table 16. Best results for CATS instances with less than 400 bids. The names of the instances are composed by the class, number of bids, number of goods, and serial number of the instance.

| Inst. | Best | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| L2_40_10_1 | 8774.7200 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L2_40_10_2 | 9229.3400 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L2_40_10_3 | 8967.4300 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L2_80_10_1 | 9828.2500 | 77.64 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L2_80_10_2 | 9786.7100 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L2_80_10_3 | 9441.1700 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L2_200_50_2 | 45785.7000 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | 92.15 | $\star$ |
| L2_200_50_3 | 49031.9000 | 87.41 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L2_400_50_1 | 46588.7410 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | 98.34 | $\star$ |
| L2_400_50_2 | 47706.0000 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L2_400_50_3 | 47819.7160 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L3_40_10_1 | 2474.2480 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L3_40_10_2 | 2682.7890 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L3_40_10_3 | 2929.2870 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L3_80_10_1 | 2862.1650 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L3_80_10_2 | 2779.9100 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L3_80_10_3 | 2938.3120 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L3_200_50_1 | 12178.8010 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L3_200_50_3 | 12612.9650 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L3_400_50_1 | 14338.1150 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L3_400_50_2 | 14747.9490 | $\star$ | $\star$ | 99.07 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L3_400_50_3 | 14495.9880 | $\star$ | $\star$ | 99.54 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L4_40_10_1 | 9543.8540 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L4_40_10_2 | 8870.7760 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L4_40_10_3 | 9249.9330 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L4_80_10_1 | 9770.0770 | $\star$ | $\star$ | 99.69 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L4_80_10_2 | 9817.6040 | $\star$ | $\star$ | 99.50 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L4_80_10_3 | 9759.7910 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L4_200_50_1 | 45191.2690 | 86.61 | $\star$ | 99.65 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L4_200_50_2 | 44275.5990 | 92.24 | $\star$ | 99.56 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L4_200_50_3 | 46496.4650 | 93.73 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L4_400_50_1 | 47748.4440 | 89.23 | $\star$ | 99.42 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L4_400_50_2 | 47988.4200 | $\star$ | $\star$ | 99.56 | 99.62 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L4_400_50_3 | 48410.5140 | $\star$ | $\star$ | 99.40 | 99.55 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L6_40_10_1 | 8791.5910 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L6_40_10_2 | 9297.1700 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L6_40_10_3 | 9217.2400 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L6_80_10_1 | 9290.9270 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L6_80_10_2 | 9836.4500 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L6_80_10_3 | 9593.9010 | $\star$ | $\star$ | 97.69 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L6_200_50_1 | 41639.9910 | 96.86 | $\star$ | 95.31 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L6_200_50_2 | 38873.5410 | 98.44 | $\star$ | 99.62 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | 99.53 | $\star$ |
| L6_200_50_3 | 40561.3300 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L6_400_50_1 | 44990.9010 | 99.12 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L6_400_50_2 | 46366.8710 | 99.45 | $\star$ | 97.48 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L6_400_50_3 | 45216.8660 | 96.18 | $\star$ | 96.83 | 95.86 | 99.92 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L7_40_10_1 | 8309.1230 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L7_40_10_2 | 9090.6580 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L7_40_10_3 | 8553.2690 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L7_80_10_1 | 9818.5880 | $\star$ | $\star$ | 99.32 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L7_80_10_2 | 9435.4580 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |
| L7_80_10_3 | 9775.6220 | $\star$ | $\star$ | 99.81 | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ | $\star$ |

Continue in next page...

Table 16: (continued).

| Inst. | Best | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| L7_200_50_1 | 28286.0100 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L7_200_50_2 | 30478.8250 | 69.46 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L7_200_50_3 | 29014.3000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L7_400_50_1 | 32505.1200 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.15 | ⋆ |
| L7_400_50_2 | 33512.5500 | ⋆ | ⋆ | 96.52 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L7_400_50_3 | 29829.2200 | ⋆ | ⋆ | 98.74 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| arbitrary_40_10_1 | 1019.7600 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| arbitrary_40_10_2 | 749.5520 | 99.98 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| arbitrary_40_10_3 | 679.8568 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| arbitrary_80_10_1 | 1038.7469 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| arbitrary_80_10_2 | 775.6040 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| arbitrary_80_10_3 | 581.9593 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| arbitrary_200_50_1 | 3007.5090 | ⋆ | ⋆ | 96.00 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.13 | ⋆ |
| arbitrary_200_50_2 | 3260.1100 | ⋆ | ⋆ | 97.54 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| arbitrary_200_50_3 | 3271.8422 | ⋆ | ⋆ | 96.60 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| arbitrary_400_50_1 | 4038.0004 | ⋆ | ⋆ | 93.09 | 90.48 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| arbitrary_400_50_2 | 3791.8860 | ⋆ | ⋆ | 93.80 | 85.02 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| arbitrary_400_50_3 | 4289.3898 | ⋆ | ⋆ | 88.70 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_40_10_1 | 10.7792 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_40_10_2 | 7.1517 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_40_10_3 | 15.6388 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_80_10_1 | 13.7825 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_80_10_2 | 1.5328 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_80_10_3 | 8.1853 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_200_50_1 | 32.2974 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_200_50_2 | 32.0509 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_200_50_3 | 23.8792 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_400_50_1 | 41.8873 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_400_50_2 | 55.8732 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_400_50_3 | 27.1398 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_40_10_1 | 4.5826 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_40_10_2 | 6.1875 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_40_10_3 | 5.3516 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_80_10_1 | 7.0575 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_80_10_2 | 5.0659 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_80_10_3 | 6.0272 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_200_50_1 | 20.2063 | 97.32 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_200_50_2 | 20.0969 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_200_50_3 | 22.1016 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_400_50_1 | 26.8886 | ⋆ | ⋆ | ⋆ | 99.54 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_400_50_2 | 22.9762 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_400_50_3 | 23.7947 | ⋆ | ⋆ | ⋆ | 99.72 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| regions_40_10_1 | 942.2510 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| regions_40_10_2 | 750.5340 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| regions_40_10_3 | 659.7730 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| regions_80_10_1 | 808.6960 | ⋆ | ⋆ | 96.58 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| regions_80_10_2 | 957.2460 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| regions_80_10_3 | 1159.6219 | ⋆ | ⋆ | 98.13 | 98.13 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| regions_200_50_1 | 3616.3098 | ⋆ | ⋆ | 97.69 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| regions_200_50_2 | 3292.0154 | ⋆ | ⋆ | 87.48 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| regions_200_50_3 | 3401.2610 | ⋆ | ⋆ | 92.77 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| regions_400_50_1 | 4177.5069 | ⋆ | ⋆ | 89.33 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| regions_400_50_2 | 3606.1991 | ⋆ | ⋆ | 91.53 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| regions_400_50_3 | 3482.6069 | ⋆ | ⋆ | 93.81 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_40_10_1 | 14.7840 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_40_10_2 | 15.1235 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_40_10_3 | 21.7354 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_80_10_1 | 22.6557 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_80_10_2 | 15.7918 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_80_10_3 | 22.8672 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_200_50_1 | 22.6139 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_200_50_2 | 53.8402 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_200_50_3 | 48.8045 | ⋆ | ⋆ | 95.26 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_400_50_1 | 58.2749 | ⋆ | ⋆ | 97.38 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_400_50_2 | 70.3185 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_400_50_3 | 43.8167 | 1.50 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |

TABLE 17. Best results for CATS instances more than 400 bids. The names of the instances are composed by the class, number of bids, number of goods, and serial number of the instance. Instances with `hard` in the name have 1024 bids and 256 goods.

| Inst. | Best | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| L2_1000_256_1 | 244098.0000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 88.40 | ⋆ |
| L2_1000_256_2 | 241158.0000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 90.88 | ⋆ |
| L2_1000_256_3 | 254988.0000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 81.73 | ⋆ |
| L2_2000_512_1 | 495448.0000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 72.84 | ⋆ |
| L2_2000_512_2 | 501810.0000 | 4.27 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 80.74 | ⋆ |
| L2_2000_512_3 | 505625.0000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 87.47 | ⋆ |
| L2_4000_1024_1 | 1000590.0000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 37.53 | ⋆ |
| L2_4000_1024_2 | 1010991.6920 | 10.66 | ⋆ | ⋆ | 100.00 | ⋆ | ⋆ | ⋆ | ⋆ | 59.42 | ⋆ |
| L2_4000_1024_3 | 996744.0000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 34.50 | ⋆ |
| L2_hard_1 | 262.5110 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 63.91 | ⋆ |
| L2_hard_2 | 456.5370 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L2_hard_3 | 317.4120 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 66.52 | ⋆ |
| L3_1000_256_1 | 64626.2530 | 75.64 | ⋆ | 99.26 | 99.60 | 99.60 | 99.60 | 99.60 | 99.56 | 96.60 | 99.50 |
| L3_1000_256_2 | 66106.1710 | 79.20 | ⋆ | 99.08 | 99.31 | 99.75 | 99.78 | 99.78 | 99.66 | 98.58 | 99.78 |
| L3_1000_256_3 | 64987.7430 | 75.45 | ⋆ | 99.68 | ⋆ | 99.84 | ⋆ | 99.81 | ⋆ | 99.79 | ⋆ |
| L3_2000_512_1 | 128004.7893 | 81.84 | ⋆ | 98.80 | 97.74 | 99.47 | 99.92 | 99.92 | 99.72 | 98.43 | 99.95 |
| L3_2000_512_2 | 132229.2010 | 93.59 | ⋆ | 99.18 | 98.78 | 99.49 | 99.71 | 99.35 | 99.78 | 97.62 | 99.69 |
| L3_2000_512_3 | 133133.1410 | 82.48 | ⋆ | 98.95 | 98.83 | 99.07 | 99.64 | 99.43 | 99.62 | 97.09 | 99.62 |
| L3_4000_1024_1 | 263970.8210 | 78.25 | ⋆ | 90.99 | 97.45 | 99.88 | 99.74 | 98.87 | 99.50 | 96.46 | 99.25 |
| L3_4000_1024_2 | 263936.8590 | 75.72 | ⋆ | 91.47 | 97.29 | 99.48 | 99.62 | 99.60 | 99.55 | 96.62 | 99.53 |
| L3_4000_1024_3 | 263404.1096 | 80.03 | ⋆ | 91.46 | 96.98 | 99.09 | 99.30 | 99.17 | 98.94 | 96.70 | 98.93 |
| L3_hard_1 | 75.4074 | 75.32 | ⋆ | 99.35 | 97.11 | 99.28 | 99.39 | 99.18 | 99.35 | 98.89 | 99.18 |
| L3_hard_2 | 34.1897 | 80.59 | 99.33 | 96.96 | 95.45 | 99.27 | ⋆ | 99.09 | 97.74 | 97.17 | 97.74 |
| L3_hard_3 | 20.8636 | 82.47 | 96.79 | 97.63 | 96.19 | 98.44 | 98.63 | 98.44 | ⋆ | 95.31 | ⋆ |
| L4_1000_256_1 | 228752.1550 | 4.41 | ⋆ | 99.52 | 99.17 | 99.85 | ⋆ | ⋆ | ⋆ | 99.67 | ⋆ |
| L4_1000_256_2 | 229601.7270 | 92.34 | ⋆ | 99.27 | 98.48 | 99.71 | ⋆ | ⋆ | ⋆ | 99.43 | ⋆ |
| L4_1000_256_3 | 229349.1950 | 2.85 | ⋆ | 99.58 | 99.70 | 99.90 | ⋆ | ⋆ | ⋆ | 99.43 | ⋆ |
| L4_2000_512_1 | 461218.2640 | 3.56 | ⋆ | 99.48 | 99.04 | 99.71 | ⋆ | ⋆ | ⋆ | 99.07 | ⋆ |
| L4_2000_512_2 | 459425.3230 | 2.75 | ⋆ | 99.83 | 98.97 | 99.73 | ⋆ | ⋆ | ⋆ | 99.32 | ⋆ |
| L4_2000_512_3 | 458536.7260 | 2.64 | ⋆ | 99.50 | 99.07 | 99.65 | ⋆ | ⋆ | ⋆ | 99.50 | ⋆ |
| L4_4000_1024_1 | 914322.9910 | 2.30 | ⋆ | 98.33 | 96.11 | 99.32 | ⋆ | ⋆ | ⋆ | 99.15 | ⋆ |
| L4_4000_1024_2 | 920786.4330 | 2.22 | ⋆ | 98.39 | 95.88 | 99.09 | ⋆ | ⋆ | ⋆ | 98.93 | ⋆ |
| L4_4000_1024_3 | 920294.1540 | 3.27 | ⋆ | 98.37 | 96.36 | 99.16 | 99.99 | 99.99 | 99.99 | 99.00 | 99.99 |
| L4_hard_1 | 290.2399 | 16.81 | ⋆ | 99.53 | 98.80 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L4_hard_2 | 383.8526 | 13.76 | ⋆ | 99.36 | 99.14 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L4_hard_3 | 282.6879 | 7.06 | ⋆ | 99.55 | 98.90 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L6_1000_256_1 | 199757.0790 | 45.81 | ⋆ | 97.22 | 99.41 | 98.23 | 98.75 | 98.32 | 98.72 | 98.22 | 98.22 |
| L6_1000_256_2 | 200559.8373 | 69.95 | ⋆ | 97.41 | 96.57 | 98.80 | 99.39 | 97.61 | 99.39 | 97.38 | 99.11 |
| L6_1000_256_3 | 201208.1706 | 3.72 | ⋆ | 99.56 | 99.15 | 98.72 | 99.96 | 98.36 | 98.92 | 97.16 | 98.36 |
| L6_2000_512_1 | 405788.3937 | 72.95 | ⋆ | 98.00 | 95.24 | 97.76 | 99.00 | 97.85 | 99.00 | 94.39 | 99.00 |
| L6_2000_512_2 | 411091.1370 | 5.22 | ⋆ | 97.83 | 94.81 | 97.77 | 98.16 | 97.22 | 98.16 | 96.02 | 97.77 |
| L6_2000_512_3 | 402472.3077 | 71.43 | ⋆ | 98.29 | 95.41 | 97.04 | 97.93 | 97.73 | 97.93 | 94.55 | 97.93 |
| L6_4000_1024_1 | 785686.0929 | 0.88 | ⋆ | 98.15 | 93.36 | 97.30 | 97.62 | 97.54 | 97.73 | 97.60 | 97.75 |
| L6_4000_1024_2 | 801026.0135 | 75.21 | ⋆ | 98.90 | 92.95 | 97.91 | 97.66 | 97.33 | 97.84 | 95.53 | 95.95 |
| L6_4000_1024_3 | 791849.8150 | 73.60 | ⋆ | 98.19 | 93.02 | 97.06 | 97.03 | 96.92 | 97.38 | 95.83 | 96.82 |
| L6_hard_1 | 377.5873 | 13.69 | ⋆ | 99.57 | 99.51 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L6_hard_2 | 330.2240 | 18.25 | ⋆ | 99.58 | 99.65 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L6_hard_3 | 446.4472 | 6.50 | ⋆ | 99.62 | 99.68 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L7_1000_256_1 | 68830.4000 | 95.17 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 86.30 | ⋆ |
| L7_1000_256_2 | 79025.8000 | 96.74 | ⋆ | ⋆ | 100.00 | ⋆ | ⋆ | ⋆ | ⋆ | 96.74 | ⋆ |
| L7_1000_256_3 | 81981.6000 | 100.00 | ⋆ | ⋆ | 100.00 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L7_2000_512_1 | 121043.0000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 97.56 | ⋆ |
| L7_2000_512_2 | 119058.0000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 93.95 | ⋆ |
| L7_2000_512_3 | 122346.0000 | 99.99 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 92.35 | ⋆ |
| L7_4000_1024_1 | 244374.0000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 90.59 | ⋆ |
| L7_4000_1024_2 | 229826.0000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.68 | ⋆ |
| L7_4000_1024_3 | 228342.0000 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 88.19 | ⋆ |
| L7_hard_1 | 233.0348 | 73.22 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.25 | ⋆ | ⋆ | ⋆ |
| L7_hard_2 | 127.4510 | 100.00 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| L7_hard_3 | 261.2782 | 83.15 | 97.72 | 97.70 | ⋆ | 99.18 | ⋆ | 99.18 | ⋆ | 95.56 | 95.56 |
| arbitrary_1000_256_1 | 17186.3016 | 70.40 | 96.39 | 93.12 | 95.91 | ⋆ | ⋆ | 96.87 | ⋆ | 94.46 | 95.53 |
| arbitrary_1000_256_2 | 15782.8217 | 6.06 | 98.02 | 96.27 | 95.63 | 98.41 | ⋆ | 98.56 | 99.40 | 96.39 | 98.03 |

Continue in next page. . .

Table 17: (continued).

| Inst. | Best | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| arbitrary_1000_256_3 | 17280.1375 | 9.31 | 98.04 | 92.55 | 97.45 | 98.69 | ⋆ | 99.28 | 99.28 | 97.28 | 97.28 |
| arbitrary_2000_512_1 | 32267.8600 | 0.17 | 96.98 | 96.15 | 93.59 | 99.74 | 98.89 | ⋆ | 99.56 | 95.35 | 96.13 |
| arbitrary_2000_512_2 | 32159.7621 | 1.56 | 95.83 | 96.42 | 94.28 | 99.97 | ⋆ | 99.39 | 99.19 | 98.02 | 97.21 |
| arbitrary_2000_512_3 | 32181.8011 | 2.52 | 95.86 | 96.79 | 97.81 | 99.54 | 99.27 | ⋆ | 99.04 | 97.77 | 98.95 |
| arbitrary_4000_1024_1 | 62694.3745 | 1.43 | 95.70 | 93.86 | 91.57 | 99.64 | 98.48 | 98.56 | ⋆ | 96.90 | 96.76 |
| arbitrary_4000_1024_2 | 61809.4598 | 0.35 | 97.64 | 93.42 | 90.90 | 97.84 | 98.90 | ⋆ | 97.92 | 96.32 | 95.92 |
| arbitrary_4000_1024_3 | 62366.9031 | 79.65 | 96.20 | 93.46 | 90.80 | 98.82 | 99.01 | 97.84 | ⋆ | 96.39 | 95.77 |
| arbitrary_hard_1 | 16412.4678 | 1.32 | 99.78 | 94.33 | 95.55 | ⋆ | 99.78 | 98.07 | 99.78 | 95.12 | 96.07 |
| arbitrary_hard_2 | 15699.7262 | 71.07 | 98.47 | 96.52 | 98.13 | ⋆ | 98.47 | 98.02 | 98.47 | 98.26 | 99.65 |
| arbitrary_hard_3 | 14954.8919 | 1.29 | 99.40 | 95.34 | 98.76 | 99.91 | 99.43 | 99.69 | 99.43 | ⋆ | ⋆ |
| matching_1000_256_1 | 724.3030 | 26.06 | ⋆ | 99.88 | 99.88 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_1000_256_2 | 731.8279 | 94.91 | ⋆ | 99.92 | 99.99 | ⋆ | ⋆ | ⋆ | ⋆ | 99.99 | ⋆ |
| matching_1000_256_3 | 912.8670 | 92.44 | ⋆ | ⋆ | 99.97 | ⋆ | ⋆ | ⋆ | ⋆ | 99.82 | ⋆ |
| matching_2000_512_1 | 669.1193 | 30.47 | ⋆ | 99.91 | 99.51 | ⋆ | ⋆ | ⋆ | ⋆ | 99.84 | ⋆ |
| matching_2000_512_2 | 1379.1604 | 92.92 | ⋆ | 99.72 | 99.48 | ⋆ | ⋆ | ⋆ | ⋆ | 99.96 | ⋆ |
| matching_2000_512_3 | 881.3102 | 24.14 | ⋆ | 99.93 | 99.35 | ⋆ | ⋆ | ⋆ | ⋆ | 99.76 | ⋆ |
| matching_4000_1024_1 | 3047.5592 | 64.56 | ⋆ | 94.35 | 98.41 | 99.98 | ⋆ | ⋆ | ⋆ | 99.82 | ⋆ |
| matching_4000_1024_2 | 2302.0147 | 38.07 | ⋆ | 94.29 | 98.19 | 100.00 | ⋆ | ⋆ | ⋆ | 99.78 | ⋆ |
| matching_4000_1024_3 | 2508.6253 | 31.33 | ⋆ | 95.11 | 98.39 | 99.94 | ⋆ | ⋆ | ⋆ | 99.87 | ⋆ |
| matching_hard_1 | 155.0591 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_hard_2 | 421.5402 | 23.49 | ⋆ | 99.96 | 99.96 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| matching_hard_3 | 323.9873 | 94.06 | ⋆ | ⋆ | 99.99 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_1000_256_1 | 57.7328 | 89.93 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_1000_256_2 | 65.7292 | 28.17 | ⋆ | ⋆ | 98.37 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_1000_256_3 | 57.4862 | 30.51 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.17 | ⋆ |
| paths_2000_512_1 | 90.3558 | 62.85 | ⋆ | 100.00 | 95.94 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| paths_2000_512_2 | 101.4873 | 52.98 | ⋆ | 99.83 | 97.01 | ⋆ | ⋆ | 100.00 | 99.98 | 99.21 | 99.98 |
| paths_2000_512_3 | 106.9681 | 57.30 | ⋆ | 99.80 | 97.50 | 99.92 | 99.87 | 99.87 | 99.87 | 99.41 | 99.87 |
| paths_4000_1024_1 | 161.5959 | 99.87 | ⋆ | 98.26 | 91.88 | 99.70 | ⋆ | ⋆ | ⋆ | 98.50 | ⋆ |
| paths_4000_1024_2 | 165.5882 | 80.80 | ⋆ | 98.52 | 93.20 | 99.47 | ⋆ | ⋆ | ⋆ | 98.79 | ⋆ |
| paths_4000_1024_3 | 150.9125 | 97.21 | ⋆ | 98.70 | 91.61 | 99.82 | ⋆ | ⋆ | ⋆ | 98.99 | ⋆ |
| regions_1000_256_1 | 16214.3571 | 74.32 | ⋆ | 95.20 | 98.99 | 98.94 | 99.36 | 98.12 | 99.36 | 98.09 | 98.89 |
| regions_1000_256_2 | 17922.5058 | 75.94 | ⋆ | 95.60 | 99.63 | 99.23 | 98.67 | 99.10 | 98.79 | 98.87 | 98.67 |
| regions_1000_256_3 | 17391.3627 | 3.78 | ⋆ | 97.09 | ⋆ | 99.54 | 99.54 | 98.18 | 99.34 | 97.85 | 99.34 |
| regions_2000_512_1 | 38262.6408 | 74.08 | ⋆ | 97.36 | 97.88 | 98.30 | 99.55 | 98.68 | 98.99 | 97.65 | 98.86 |
| regions_2000_512_2 | 32274.1576 | 62.09 | ⋆ | 95.82 | 95.49 | 99.77 | 98.64 | 97.89 | 98.60 | 96.23 | 97.45 |
| regions_2000_512_3 | 37199.7468 | 1.20 | ⋆ | 96.77 | 98.55 | 99.21 | 99.59 | 98.49 | 99.50 | 97.50 | 98.95 |
| regions_4000_1024_1 | 65807.6502 | 0.33 | 99.97 | 94.38 | 96.23 | 99.56 | 99.28 | 97.14 | ⋆ | 96.09 | 96.67 |
| regions_4000_1024_2 | 65628.9304 | 2.72 | 99.70 | 95.57 | 97.09 | 99.91 | ⋆ | 98.25 | 99.30 | 98.75 | 98.97 |
| regions_4000_1024_3 | 64800.3099 | 5.43 | ⋆ | 95.15 | 95.79 | 99.06 | 99.04 | 96.26 | 99.42 | 94.39 | 95.07 |
| regions_hard_1 | 15336.4868 | 72.73 | ⋆ | 94.01 | 98.24 | 99.82 | 99.82 | 99.22 | 99.82 | 98.55 | 99.82 |
| regions_hard_2 | 17988.3370 | 70.19 | ⋆ | 94.17 | 99.48 | 99.48 | 99.74 | 98.55 | 99.74 | 99.74 | 99.48 |
| regions_hard_3 | 16777.2344 | 72.67 | ⋆ | 93.30 | 98.03 | 99.17 | 99.66 | 98.75 | 99.66 | 97.94 | 98.35 |
| scheduling_1000_256_1 | 44.9038 | 22.99 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_1000_256_2 | 42.5548 | 20.98 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_1000_256_3 | 87.6889 | 11.41 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_2000_512_1 | 40.9792 | 25.74 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_2000_512_2 | 58.2106 | 3.63 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_2000_512_3 | 48.2352 | 1.32 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_4000_1024_1 | 28.3994 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_4000_1024_2 | 45.9743 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_4000_1024_3 | 36.6752 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_hard_1 | 168.4070 | 98.37 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_hard_2 | 1219.4075 | 100.00 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| scheduling_hard_3 | 4812.6430 | 56.44 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |

TABLE 18. Best results for LG 1000/500 instances.

| Inst. | Best | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| in101 | 72724.6180 | 37.66 | 92.27 | 92.27 | 96.03 | ⋆ | ⋆ | ⋆ | ⋆ | 95.02 | 98.63 |
| in102 | 72518.2220 | 45.17 | 98.03 | 96.72 | 98.22 | ⋆ | ⋆ | ⋆ | ⋆ | 97.44 | 99.82 |
| in103 | 72129.5000 | 40.67 | 96.64 | 95.13 | 96.76 | ⋆ | ⋆ | 97.41 | ⋆ | ⋆ | 98.43 |
| in104 | 72709.6470 | 65.30 | 98.04 | 94.46 | 97.42 | ⋆ | ⋆ | ⋆ | ⋆ | 92.45 | ⋆ |
| in105 | 75646.1406 | 39.96 | 89.05 | 90.91 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 94.99 | ⋆ |

Continue in next page...

Table 18: (continued).

| Inst. | Best | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| in106 | 71258.6130 | 51.23 | 89.77 | 93.23 | 94.31 | ⋆ | ⋆ | ⋆ | ⋆ | 94.49 | ⋆ |
| in107 | 69713.4030 | 38.64 | 98.38 | 98.38 | 99.24 | ⋆ | ⋆ | 99.55 | ⋆ | ⋆ | ⋆ |
| in108 | 75813.2109 | 11.33 | 98.39 | 99.12 | ⋆ | 99.95 | ⋆ | 99.12 | 99.95 | 99.31 | 99.30 |
| in109 | 69475.8950 | 38.47 | 91.99 | 95.09 | 95.34 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in110 | 68295.2890 | 16.29 | ⋆ | 92.75 | 99.79 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in111 | 75133.2900 | 42.87 | 96.74 | 95.16 | 97.12 | ⋆ | ⋆ | ⋆ | ⋆ | 95.53 | 97.12 |
| in112 | 71342.4830 | 60.02 | 99.25 | 94.80 | 99.81 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in113 | 73365.8906 | 53.84 | 92.73 | 96.02 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.43 | ⋆ |
| in114 | 69224.7656 | 13.31 | 94.35 | 94.58 | ⋆ | ⋆ | 98.58 | 99.56 | 99.56 | 96.04 | 96.04 |
| in115 | 70221.5610 | 48.33 | 94.85 | 93.15 | 96.06 | ⋆ | ⋆ | 99.55 | ⋆ | 95.95 | 96.95 |
| in116 | 70032.4609 | 48.56 | 98.32 | 93.80 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.32 | 98.32 |
| in117 | 69982.8330 | 59.34 | 94.96 | 99.92 | 99.92 | ⋆ | ⋆ | ⋆ | ⋆ | 95.33 | 98.99 |
| in118 | 72160.9870 | 57.56 | 95.02 | 93.06 | 97.21 | ⋆ | ⋆ | ⋆ | ⋆ | 95.36 | 95.36 |
| in119 | 67038.4297 | 58.44 | 96.86 | ⋆ | ⋆ | ⋆ | ⋆ | 98.36 | ⋆ | 98.27 | 98.27 |
| in120 | 75514.9300 | 58.27 | 98.85 | 93.41 | 99.95 | ⋆ | ⋆ | 99.13 | 99.13 | 97.67 | 98.87 |
| in121 | 67639.1250 | 34.44 | 96.47 | 94.24 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 96.34 | 96.34 |
| in122 | 69546.2730 | 40.98 | 96.73 | 98.24 | 98.24 | ⋆ | ⋆ | 99.97 | ⋆ | 95.69 | ⋆ |
| in123 | 70618.3130 | 49.50 | 92.25 | 94.96 | 99.97 | ⋆ | ⋆ | ⋆ | 99.97 | 98.78 | 99.93 |
| in124 | 71686.0469 | 39.54 | 97.28 | 99.72 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 96.63 | 99.72 |
| in125 | 69233.1220 | 51.98 | 95.79 | 95.79 | 97.41 | ⋆ | ⋆ | ⋆ | ⋆ | 98.14 | ⋆ |
| in126 | 70671.7700 | 9.53 | 98.45 | 93.05 | 98.61 | ⋆ | ⋆ | 98.61 | ⋆ | 95.98 | 96.62 |
| in127 | 69273.3203 | 42.94 | 98.39 | 92.27 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.74 | ⋆ |
| in128 | 72179.4310 | 17.30 | 94.35 | 90.70 | 98.32 | ⋆ | ⋆ | ⋆ | ⋆ | 95.27 | 96.20 |
| in129 | 65751.6490 | 37.24 | 97.51 | 97.51 | 97.59 | ⋆ | ⋆ | ⋆ | ⋆ | 97.51 | 97.51 |
| in130 | 71075.3000 | 48.78 | 97.39 | 97.90 | 97.90 | ⋆ | ⋆ | 99.14 | ⋆ | 96.04 | 97.90 |
| in131 | 71177.9062 | 2.62 | 95.49 | 99.62 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 96.39 | ⋆ |
| in132 | 75510.0469 | 43.34 | 96.88 | 99.90 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in133 | 71253.5610 | 54.48 | 97.85 | 94.16 | 99.35 | ⋆ | ⋆ | 99.35 | ⋆ | 94.95 | 97.67 |
| in134 | 75781.7490 | 46.70 | 96.61 | 91.22 | 98.73 | ⋆ | ⋆ | ⋆ | ⋆ | 97.39 | ⋆ |
| in135 | 72138.1172 | 2.42 | 95.49 | 90.72 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in136 | 68903.0938 | 43.37 | 94.79 | 96.29 | ⋆ | ⋆ | ⋆ | 99.86 | 99.86 | 96.61 | 99.04 |
| in137 | 70072.0469 | 48.96 | ⋆ | 90.56 | ⋆ | ⋆ | ⋆ | 99.99 | ⋆ | ⋆ | ⋆ |
| in138 | 71989.6330 | 28.25 | 97.43 | 99.24 | 99.24 | 99.24 | ⋆ | 99.24 | ⋆ | 99.24 | 97.71 |
| in139 | 72840.3940 | 35.02 | 94.24 | 92.53 | 98.79 | ⋆ | ⋆ | ⋆ | ⋆ | 96.13 | 98.94 |
| in140 | 73665.2310 | 43.72 | ⋆ | 92.15 | 92.42 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in141 | 69605.0770 | 40.43 | 98.91 | 96.15 | 99.67 | ⋆ | ⋆ | ⋆ | ⋆ | 95.42 | 98.91 |
| in142 | 74777.9850 | 49.90 | 97.26 | 94.59 | 96.20 | ⋆ | ⋆ | ⋆ | ⋆ | 97.52 | 97.52 |
| in143 | 69699.0547 | 34.12 | 95.14 | 98.81 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.19 | 98.89 |
| in144 | 73197.0730 | 49.56 | 94.95 | 93.48 | 99.03 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in145 | 73695.0150 | 39.38 | 96.88 | 92.77 | 96.25 | ⋆ | ⋆ | 97.81 | ⋆ | ⋆ | 97.80 |
| in146 | 73746.9375 | 38.30 | 95.29 | 93.65 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 97.29 | 97.29 |
| in147 | 65878.3020 | 58.53 | 95.28 | 97.17 | 97.17 | ⋆ | ⋆ | ⋆ | ⋆ | 94.88 | 94.88 |
| in148 | 72116.9690 | 51.94 | 96.01 | 95.66 | 98.84 | 99.81 | ⋆ | 98.84 | ⋆ | 99.81 | 99.81 |
| in149 | 70800.1800 | 46.53 | 97.27 | 95.68 | 98.61 | ⋆ | ⋆ | 99.30 | ⋆ | 99.02 | ⋆ |
| in150 | 72839.4240 | 46.46 | 94.35 | 93.20 | 98.91 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in151 | 68834.5010 | 45.83 | 99.99 | 97.90 | 99.13 | ⋆ | ⋆ | ⋆ | ⋆ | 98.85 | 99.75 |
| in152 | 76224.7812 | 41.04 | 93.71 | 93.62 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 97.94 | 97.94 |
| in153 | 70110.7650 | 43.46 | 99.49 | 96.81 | 99.49 | ⋆ | ⋆ | 99.60 | ⋆ | 96.54 | 98.00 |
| in154 | 69215.5240 | 7.16 | 94.14 | 96.66 | 98.48 | ⋆ | 99.27 | 99.27 | 99.27 | 99.09 | 99.27 |
| in155 | 74936.7730 | 36.64 | 96.51 | 96.26 | 97.22 | ⋆ | ⋆ | 99.75 | ⋆ | 99.75 | ⋆ |
| in156 | 69704.1300 | 50.74 | 93.01 | 99.24 | 99.24 | ⋆ | ⋆ | ⋆ | ⋆ | 96.64 | 96.64 |
| in157 | 73934.8438 | 33.75 | 91.20 | 93.38 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 92.71 | ⋆ |
| in158 | 69489.5430 | 47.97 | ⋆ | 92.69 | 97.71 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 95.13 |
| in159 | 71091.8047 | 55.58 | 96.38 | 95.46 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.53 | 98.53 |
| in160 | 70606.9180 | 46.45 | 96.31 | 99.48 | 99.48 | ⋆ | ⋆ | ⋆ | ⋆ | 98.61 | 98.61 |
| in161 | 66266.3710 | 15.81 | 92.56 | 93.91 | 98.53 | ⋆ | 99.34 | ⋆ | ⋆ | 97.88 | ⋆ |
| in162 | 74720.7940 | 54.27 | 93.32 | 95.58 | 97.44 | 99.44 | 99.44 | ⋆ | ⋆ | 99.44 | 99.44 |
| in163 | 64976.9910 | 46.23 | 98.63 | 98.45 | 99.06 | ⋆ | 99.86 | 99.86 | 99.86 | 99.06 | 99.06 |
| in164 | 67950.6230 | 43.67 | 93.99 | 91.64 | 99.41 | ⋆ | ⋆ | ⋆ | ⋆ | 98.38 | 98.38 |
| in165 | 70361.9531 | 39.37 | 95.13 | 95.97 | ⋆ | ⋆ | ⋆ | 98.61 | ⋆ | 97.19 | 97.19 |
| in166 | 71460.8930 | 34.20 | 92.35 | 95.05 | 97.99 | 99.80 | ⋆ | 99.45 | 99.80 | 97.56 | 97.83 |
| in167 | 74523.7656 | 26.61 | ⋆ | 96.58 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 96.86 | 96.86 |
| in168 | 72097.3210 | 43.31 | 97.37 | 96.68 | 96.86 | ⋆ | ⋆ | 99.54 | ⋆ | ⋆ | ⋆ |
| in169 | 71827.3400 | 45.18 | 96.43 | 97.32 | 98.45 | ⋆ | ⋆ | 98.62 | ⋆ | 95.21 | ⋆ |
| in170 | 74564.7490 | 42.64 | 92.70 | 92.51 | 95.80 | ⋆ | ⋆ | 96.46 | ⋆ | 90.03 | 95.15 |
| in171 | 71279.4840 | 37.03 | 96.53 | 94.32 | 97.33 | ⋆ | ⋆ | 98.48 | ⋆ | 98.48 | 98.45 |
| in172 | 70361.8070 | 3.68 | 99.57 | 93.91 | 96.82 | ⋆ | ⋆ | 99.57 | ⋆ | 97.41 | 98.66 |

Continue in next page...

Table 18: (continued).

| Inst. | Best | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| in173 | 73677.2030 | 57.65 | 96.29 | 93.20 | 99.78 | ⋆ | ⋆ | ⋆ | ⋆ | 94.10 | 97.49 |
| in174 | 73523.6094 | 44.59 | 96.31 | 92.89 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 96.48 | 95.46 |
| in175 | 72924.8740 | 49.45 | 97.54 | 91.49 | 97.26 | ⋆ | 99.67 | 99.67 | 99.67 | 99.67 | 99.67 |
| in176 | 67761.4830 | 38.10 | 97.51 | 95.42 | 99.35 | ⋆ | ⋆ | ⋆ | ⋆ | 98.33 | 99.43 |
| in177 | 70187.1540 | 49.13 | 94.27 | 95.49 | 98.94 | ⋆ | ⋆ | ⋆ | ⋆ | 98.88 | 98.94 |
| in178 | 70833.3720 | 53.70 | 90.85 | 92.84 | 95.71 | ⋆ | ⋆ | ⋆ | ⋆ | 94.42 | 95.20 |
| in179 | 72205.2980 | 42.51 | 96.80 | 95.57 | 96.60 | ⋆ | ⋆ | 98.97 | ⋆ | 96.60 | ⋆ |
| in180 | 70513.3520 | 54.50 | 93.16 | 94.07 | 96.53 | ⋆ | ⋆ | ⋆ | ⋆ | 96.26 | 96.31 |
| in181 | 72238.0859 | 37.33 | 95.06 | 97.16 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 96.76 | 99.07 |
| in182 | 71645.0312 | 37.70 | 97.82 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.80 | 98.80 |
| in183 | 71520.4688 | 37.89 | 98.38 | 93.86 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.57 | 99.57 |
| in184 | 74377.5380 | 1.74 | 92.64 | 87.92 | 94.41 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in185 | 73714.9531 | 47.24 | ⋆ | 94.44 | ⋆ | ⋆ | ⋆ | 99.52 | ⋆ | 99.52 | ⋆ |
| in186 | 70736.2480 | 47.66 | 97.98 | 94.98 | 98.72 | ⋆ | ⋆ | ⋆ | ⋆ | 97.98 | 97.98 |
| in187 | 70166.3660 | 31.20 | 95.26 | 94.28 | 97.34 | ⋆ | ⋆ | ⋆ | ⋆ | 98.55 | 98.55 |
| in188 | 70485.1950 | 40.11 | 93.17 | 95.47 | 96.52 | ⋆ | ⋆ | 98.86 | ⋆ | 99.49 | ⋆ |
| in189 | 69786.0220 | 38.77 | 95.35 | 96.84 | 98.82 | ⋆ | ⋆ | ⋆ | ⋆ | 98.54 | ⋆ |
| in190 | 73765.2090 | 38.54 | 97.07 | 97.07 | 98.60 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.72 |
| in191 | 72587.0780 | 8.24 | 99.65 | 97.87 | 98.63 | ⋆ | ⋆ | 99.65 | ⋆ | ⋆ | ⋆ |
| in192 | 71156.8280 | 34.93 | 93.02 | 94.22 | 99.45 | ⋆ | ⋆ | ⋆ | ⋆ | 99.61 | 99.61 |
| in193 | 72526.4688 | 33.95 | 97.21 | 94.22 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 97.46 | 97.46 |
| in194 | 75803.5156 | 47.87 | 94.14 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.91 | 99.91 |
| in195 | 69066.8672 | 29.99 | 91.41 | 96.21 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 96.25 | 96.25 |
| in196 | 69776.2220 | 51.81 | 98.39 | 98.47 | 99.91 | ⋆ | ⋆ | 99.91 | ⋆ | 98.70 | 97.77 |
| in197 | 68457.8040 | 55.49 | ⋆ | ⋆ | 98.33 | ⋆ | ⋆ | ⋆ | ⋆ | 97.41 | 97.41 |
| in198 | 73474.3830 | 41.26 | 98.84 | 92.37 | 97.19 | ⋆ | ⋆ | ⋆ | ⋆ | 97.19 | 97.19 |
| in199 | 70955.9130 | 37.03 | 93.84 | 95.59 | 99.98 | ⋆ | ⋆ | 99.98 | ⋆ | 98.21 | 98.34 |
| in200 | 76803.1830 | 46.02 | 95.19 | 95.17 | 98.09 | ⋆ | ⋆ | 98.88 | ⋆ | ⋆ | ⋆ |

TABLE 19. Best results for LG 1000/1000 instances.

| Inst. | Best | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| in201 | 81557.7578 | 45.28 | 94.10 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.71 | 96.02 |
| in202 | 90708.1406 | 38.91 | 98.60 | 93.30 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.96 | ⋆ |
| in203 | 86239.2266 | 7.67 | 95.45 | 93.96 | ⋆ | ⋆ | ⋆ | 99.12 | ⋆ | 97.69 | 97.69 |
| in204 | 87075.4453 | 38.39 | 94.14 | 94.39 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 95.49 | 98.62 |
| in205 | 86515.9510 | 34.40 | 93.59 | 92.44 | 97.11 | ⋆ | ⋆ | ⋆ | ⋆ | 94.14 | 96.80 |
| in206 | 91518.9640 | 19.12 | 94.93 | 93.41 | 94.93 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in207 | 93129.2900 | 27.22 | ⋆ | 97.75 | 99.99 | ⋆ | ⋆ | 99.99 | ⋆ | 94.91 | 94.91 |
| in208 | 94904.6953 | 25.73 | 88.80 | 90.18 | ⋆ | ⋆ | ⋆ | 96.71 | ⋆ | 96.71 | 96.71 |
| in209 | 87268.9650 | 47.58 | 98.88 | 93.37 | 99.41 | ⋆ | ⋆ | 99.41 | ⋆ | 98.88 | 96.83 |
| in210 | 89962.4062 | 39.71 | 96.64 | 95.73 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.38 | ⋆ |
| in211 | 84913.6840 | 55.07 | 93.20 | 92.87 | 99.54 | ⋆ | ⋆ | ⋆ | ⋆ | 97.60 | 98.48 |
| in212 | 90778.2188 | 40.06 | 96.81 | 91.38 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.97 | 98.97 |
| in213 | 85369.1850 | 34.71 | 95.81 | 97.87 | 97.87 | ⋆ | ⋆ | ⋆ | ⋆ | 98.97 | 98.97 |
| in214 | 85181.6090 | 39.16 | 97.34 | 96.19 | 99.58 | ⋆ | ⋆ | ⋆ | ⋆ | 99.58 | ⋆ |
| in215 | 91531.7031 | 46.31 | 95.42 | 93.46 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.56 | 97.91 |
| in216 | 91580.9800 | 48.61 | ⋆ | 93.53 | 94.72 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in217 | 86962.9270 | 52.45 | 97.72 | 93.77 | 98.33 | ⋆ | ⋆ | ⋆ | ⋆ | 97.72 | 99.92 |
| in218 | 94965.2109 | 45.14 | 90.34 | 91.55 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in219 | 93586.4380 | 46.99 | 90.94 | 96.02 | 96.02 | ⋆ | ⋆ | ⋆ | ⋆ | 96.08 | 96.08 |
| in220 | 89792.9219 | 44.44 | 97.87 | 96.82 | ⋆ | ⋆ | ⋆ | 98.48 | ⋆ | 98.63 | 98.63 |
| in221 | 87410.7800 | 41.62 | ⋆ | 93.56 | 97.23 | ⋆ | ⋆ | ⋆ | ⋆ | 96.00 | 96.23 |
| in222 | 89905.5391 | 45.82 | 94.77 | 90.80 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in223 | 83045.4297 | 40.13 | 96.04 | 88.95 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 94.30 | 94.71 |
| in224 | 87105.2770 | 49.10 | 96.86 | 98.39 | 99.92 | ⋆ | ⋆ | ⋆ | ⋆ | 97.13 | 97.40 |
| in225 | 89430.1094 | 38.68 | 95.90 | 91.21 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in226 | 88176.1220 | 34.96 | 91.75 | 90.75 | 95.92 | ⋆ | ⋆ | 95.09 | ⋆ | 95.92 | 95.92 |
| in227 | 92613.3710 | 44.80 | 96.95 | 95.57 | 98.94 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in228 | 92684.0781 | 56.28 | 96.70 | 96.70 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.86 | 95.33 |
| in229 | 90468.1420 | 49.34 | 96.50 | 91.38 | 96.75 | ⋆ | ⋆ | ⋆ | ⋆ | 96.76 | 96.76 |
| in230 | 91559.1562 | 48.44 | 96.66 | 94.13 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 97.74 | 97.74 |
| in231 | 101458.6094 | 40.11 | 93.07 | 88.09 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in232 | 87270.8630 | 17.55 | 95.18 | 91.66 | 99.45 | ⋆ | ⋆ | ⋆ | ⋆ | 92.66 | ⋆ |

Continue in next page...

Table 19: (continued).

| Inst. | Best | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| in233 | 86151.8980 | 39.85 | 96.84 | 94.09 | 98.81 | ⋆ | ⋆ | ⋆ | ⋆ | 96.91 | 97.09 |
| in234 | 88874.3359 | 49.60 | 98.17 | 92.70 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 96.84 | 96.84 |
| in235 | 93246.5700 | 38.90 | ⋆ | 89.98 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 93.01 | 97.24 |
| in236 | 87876.7891 | 38.94 | 98.10 | 91.59 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 95.25 | 96.84 |
| in237 | 87616.0450 | 54.09 | 96.48 | 94.61 | 98.30 | ⋆ | ⋆ | ⋆ | ⋆ | 97.16 | 99.78 |
| in238 | 87004.0781 | 49.72 | 98.22 | 90.70 | ⋆ | ⋆ | ⋆ | 99.57 | ⋆ | 98.70 | ⋆ |
| in239 | 81435.3020 | 41.92 | 99.86 | 92.88 | 99.86 | ⋆ | ⋆ | ⋆ | ⋆ | 98.41 | 98.41 |
| in240 | 86608.4120 | 45.38 | 98.11 | 98.61 | 98.61 | ⋆ | ⋆ | ⋆ | ⋆ | 95.04 | 98.90 |
| in241 | 89961.1641 | 39.00 | 98.80 | 92.63 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.80 | 98.80 |
| in242 | 92480.5420 | 35.73 | 90.80 | 91.44 | 92.68 | ⋆ | ⋆ | ⋆ | ⋆ | 95.12 | 96.56 |
| in243 | 91839.5970 | 37.24 | 99.91 | 91.99 | 99.91 | ⋆ | ⋆ | ⋆ | ⋆ | 96.47 | 96.47 |
| in244 | 91029.7940 | 42.40 | 95.61 | 92.89 | 98.11 | ⋆ | ⋆ | 98.11 | ⋆ | 97.04 | 97.04 |
| in245 | 90590.5630 | 34.27 | 95.38 | 96.10 | 96.10 | ⋆ | ⋆ | ⋆ | ⋆ | 94.46 | 99.06 |
| in246 | 87158.2344 | 24.83 | 99.39 | ⋆ | ⋆ | ⋆ | ⋆ | 99.17 | ⋆ | 99.39 | 99.17 |
| in247 | 89044.3828 | 45.42 | 96.32 | 96.01 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.54 | 99.54 |
| in248 | 93058.1406 | 57.39 | 91.53 | 92.92 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 95.73 | 95.73 |
| in249 | 95169.5190 | 62.17 | 93.98 | 93.98 | 98.01 | ⋆ | ⋆ | ⋆ | ⋆ | 96.22 | 96.22 |
| in250 | 93775.8359 | 48.37 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.82 | 98.82 |
| in251 | 88734.0770 | 43.94 | 92.56 | 92.35 | 96.09 | ⋆ | ⋆ | 96.42 | ⋆ | 96.11 | 96.11 |
| in252 | 89504.9220 | 53.90 | 93.49 | 98.03 | 98.03 | ⋆ | ⋆ | ⋆ | ⋆ | 99.86 | 99.86 |
| in253 | 88253.3125 | 24.40 | 95.78 | 96.70 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 96.87 | ⋆ |
| in254 | 85897.5010 | 31.00 | 96.01 | 96.37 | 96.37 | ⋆ | ⋆ | ⋆ | ⋆ | 98.85 | ⋆ |
| in255 | 89368.1990 | 37.11 | 94.69 | 94.32 | 98.13 | ⋆ | ⋆ | 97.74 | ⋆ | 98.67 | 98.67 |
| in256 | 89253.2656 | 38.86 | 93.03 | 92.12 | ⋆ | ⋆ | ⋆ | 96.74 | ⋆ | ⋆ | 95.03 |
| in257 | 88605.5950 | 12.67 | 96.54 | 94.88 | 99.49 | ⋆ | ⋆ | ⋆ | ⋆ | 97.23 | 99.17 |
| in258 | 85183.9110 | 44.59 | 99.33 | 97.74 | 98.65 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in259 | 95397.3516 | 37.58 | ⋆ | 87.77 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 93.80 | 93.80 |
| in260 | 90407.2050 | 42.48 | 99.25 | 92.46 | 96.03 | ⋆ | ⋆ | ⋆ | ⋆ | 99.38 | ⋆ |
| in261 | 89790.1900 | 46.72 | ⋆ | 92.80 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 97.30 | 97.30 |
| in262 | 88470.1100 | 50.02 | ⋆ | 92.98 | 99.03 | ⋆ | ⋆ | ⋆ | ⋆ | 96.68 | ⋆ |
| in263 | 93087.8530 | 37.55 | 94.59 | 94.24 | 97.35 | ⋆ | ⋆ | ⋆ | ⋆ | 98.98 | 98.98 |
| in264 | 86498.9141 | 48.56 | 97.86 | 91.86 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.00 | 99.00 |
| in265 | 83621.1700 | 41.51 | 95.48 | 97.91 | 98.94 | ⋆ | ⋆ | 98.47 | ⋆ | ⋆ | 99.16 |
| in266 | 90038.9920 | 31.15 | 96.12 | 94.84 | 98.48 | ⋆ | ⋆ | ⋆ | ⋆ | 98.50 | 98.50 |
| in267 | 91438.2109 | 24.48 | 99.40 | 92.66 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in268 | 89482.2790 | 41.41 | 97.93 | 93.04 | 99.78 | ⋆ | ⋆ | ⋆ | ⋆ | 98.80 | 98.80 |
| in269 | 83546.6830 | 48.56 | 99.19 | 96.77 | 99.88 | ⋆ | ⋆ | ⋆ | ⋆ | 99.46 | 99.46 |
| in270 | 87509.4062 | 34.87 | 97.20 | 92.81 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 95.73 | 95.73 |
| in271 | 85951.6810 | 42.83 | 95.87 | 93.42 | 97.72 | ⋆ | ⋆ | ⋆ | ⋆ | 98.51 | 98.51 |
| in272 | 88642.8220 | 49.07 | 92.82 | 95.86 | 97.28 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in273 | 87909.9070 | 39.27 | 99.20 | 95.06 | 99.96 | ⋆ | ⋆ | 99.21 | ⋆ | ⋆ | ⋆ |
| in274 | 83417.7890 | 45.77 | 98.89 | 93.02 | 99.18 | ⋆ | ⋆ | 98.89 | ⋆ | 98.33 | 98.33 |
| in275 | 89915.1500 | 37.12 | 98.05 | 94.19 | 99.17 | ⋆ | ⋆ | 99.57 | ⋆ | 98.79 | 98.79 |
| in276 | 86626.4375 | 50.65 | 99.36 | 99.40 | ⋆ | ⋆ | ⋆ | 99.78 | ⋆ | 97.18 | 99.36 |
| in277 | 88537.7270 | 37.49 | 96.56 | 98.09 | 98.79 | ⋆ | ⋆ | 98.79 | ⋆ | 96.58 | 96.58 |
| in278 | 91326.9531 | 52.88 | 95.55 | 96.72 | ⋆ | ⋆ | ⋆ | 99.28 | ⋆ | 99.28 | 99.28 |
| in279 | 87058.9800 | 46.83 | ⋆ | 89.91 | 96.27 | ⋆ | ⋆ | 98.88 | ⋆ | 97.82 | 98.45 |
| in280 | 86529.5938 | 38.92 | 97.14 | 93.11 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.46 | 99.46 |
| in281 | 88470.4141 | 53.98 | 96.51 | 95.42 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.75 | 99.75 |
| in282 | 88985.3290 | 46.25 | 92.23 | 91.56 | 96.27 | ⋆ | ⋆ | ⋆ | ⋆ | 95.57 | 96.27 |
| in283 | 88915.6590 | 49.94 | 95.47 | 96.33 | 96.33 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in284 | 88241.9750 | 39.20 | 96.30 | 96.86 | 96.86 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 97.14 |
| in285 | 85953.2490 | 45.59 | 96.57 | 93.35 | 97.89 | ⋆ | ⋆ | ⋆ | ⋆ | 99.16 | 99.16 |
| in286 | 88323.4844 | 57.41 | 92.98 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 92.53 | 92.31 |
| in287 | 91652.7400 | 32.42 | 99.46 | 88.37 | 99.46 | ⋆ | ⋆ | ⋆ | ⋆ | 93.58 | ⋆ |
| in288 | 85639.0090 | 41.68 | 95.93 | 96.81 | 97.90 | ⋆ | ⋆ | ⋆ | ⋆ | 96.81 | 96.17 |
| in289 | 86032.8140 | 49.01 | 96.03 | 91.94 | 96.03 | ⋆ | ⋆ | ⋆ | ⋆ | 97.12 | 97.48 |
| in290 | 92103.2070 | 42.87 | 95.79 | 90.63 | 96.06 | ⋆ | ⋆ | ⋆ | ⋆ | 94.55 | 95.24 |
| in291 | 94188.2910 | 59.11 | 92.98 | 95.55 | 96.30 | ⋆ | ⋆ | ⋆ | ⋆ | 94.64 | 94.64 |
| in292 | 94063.9650 | 57.29 | 97.14 | 95.96 | 96.56 | ⋆ | ⋆ | ⋆ | ⋆ | 97.90 | ⋆ |
| in293 | 85810.6210 | 51.78 | 98.18 | 95.25 | 98.82 | ⋆ | ⋆ | ⋆ | ⋆ | 97.04 | ⋆ |
| in294 | 91167.3160 | 49.30 | 94.62 | 91.92 | 96.80 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in295 | 89267.5156 | 34.74 | 94.01 | 93.08 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 95.56 |
| in296 | 90000.2970 | 22.43 | 98.55 | 93.35 | 98.55 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in297 | 89725.9360 | 27.45 | ⋆ | 94.05 | 94.84 | ⋆ | ⋆ | ⋆ | ⋆ | 97.43 | 97.43 |
| in298 | 89166.7422 | 47.96 | 98.65 | 93.52 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.77 |
| in299 | 92218.6094 | 41.60 | 98.03 | 95.49 | ⋆ | ⋆ | ⋆ | 99.53 | ⋆ | 93.10 | 93.10 |

Continue in next page...

Table 19: (continued).

| Inst. | Best | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| in300 | 88373.3281 | 48.68 | 97.91 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.30 | 96.75 |

TABLE 20. Best results for LG 1500/1500 instances.

| Inst. | Best | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| in601 | 108800.4450 | 58.25 | 95.76 | 91.03 | 96.77 | ⋆ | ⋆ | 98.18 | ⋆ | 97.43 | 97.43 |
| in602 | 105611.4760 | 24.41 | 93.92 | 92.95 | 95.78 | ⋆ | ⋆ | ⋆ | ⋆ | 94.12 | 94.12 |
| in603 | 105121.0220 | 39.04 | 92.40 | 88.06 | 92.54 | ⋆ | ⋆ | 97.57 | ⋆ | ⋆ | ⋆ |
| in604 | 107733.8050 | 50.52 | 96.29 | 96.13 | 96.29 | 98.96 | 98.96 | 98.96 | ⋆ | 98.04 | 98.04 |
| in605 | 109840.9840 | 52.38 | 93.23 | 92.38 | 94.98 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in606 | 107113.0670 | 37.26 | 92.47 | 97.42 | 98.23 | ⋆ | ⋆ | ⋆ | ⋆ | 93.83 | 93.83 |
| in607 | 113180.2840 | 43.74 | 90.23 | 93.54 | 93.54 | ⋆ | ⋆ | ⋆ | ⋆ | 91.09 | ⋆ |
| in608 | 105266.1070 | 50.50 | 96.26 | 88.48 | 97.88 | ⋆ | ⋆ | ⋆ | ⋆ | 99.07 | 99.07 |
| in609 | 109472.3320 | 3.23 | 96.71 | 90.87 | 95.77 | ⋆ | ⋆ | ⋆ | ⋆ | 94.18 | 95.55 |
| in610 | 113716.9650 | 32.91 | 93.89 | 88.04 | 95.86 | ⋆ | ⋆ | 98.04 | ⋆ | 95.87 | 98.04 |
| in611 | 106666.3438 | 10.31 | 94.57 | 88.69 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.76 | 98.76 |
| in612 | 109796.7400 | 54.31 | ⋆ | 96.59 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 96.91 | 94.91 |
| in613 | 107980.1570 | 71.82 | 93.49 | 86.40 | 93.09 | ⋆ | ⋆ | ⋆ | ⋆ | 96.58 | ⋆ |
| in614 | 108364.5859 | 49.66 | ⋆ | 89.98 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in615 | 110508.8281 | 37.36 | 97.15 | 86.14 | ⋆ | ⋆ | ⋆ | 98.92 | ⋆ | 92.40 | 92.40 |
| in616 | 109740.4922 | 44.02 | 88.30 | 91.67 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 95.39 | 96.64 |
| in617 | 113302.4340 | 45.99 | 92.27 | 91.02 | 93.78 | ⋆ | ⋆ | ⋆ | ⋆ | 91.75 | 95.29 |
| in618 | 111385.0810 | 47.03 | 94.81 | 88.56 | 99.93 | ⋆ | ⋆ | 99.58 | ⋆ | 95.53 | 99.57 |
| in619 | 107571.5930 | 43.20 | 97.72 | 90.46 | 94.97 | ⋆ | ⋆ | ⋆ | ⋆ | 97.72 | ⋆ |
| in620 | 110937.9750 | 59.54 | 92.13 | 93.65 | 95.07 | ⋆ | ⋆ | 97.96 | ⋆ | 97.96 | 96.75 |
| in621 | 106133.8500 | 41.62 | ⋆ | 93.40 | 93.40 | ⋆ | ⋆ | ⋆ | ⋆ | 98.14 | 98.14 |
| in622 | 107551.7370 | 55.58 | 91.12 | 94.14 | 98.49 | ⋆ | ⋆ | ⋆ | ⋆ | 96.71 | 97.53 |
| in623 | 109487.0290 | 42.38 | 94.77 | 94.57 | 94.99 | ⋆ | ⋆ | ⋆ | ⋆ | 96.90 | 96.90 |
| in624 | 104386.9790 | 48.61 | 92.76 | 92.27 | 95.69 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in625 | 109065.3594 | 43.83 | 96.90 | 88.55 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 94.30 | 97.57 |
| in626 | 114704.0340 | 50.12 | 89.36 | 88.28 | 96.60 | ⋆ | ⋆ | ⋆ | ⋆ | 97.77 | 92.88 |
| in627 | 108846.2344 | 37.55 | 91.65 | 95.31 | ⋆ | ⋆ | ⋆ | 99.17 | ⋆ | 99.17 | 99.17 |
| in628 | 108169.6953 | 42.91 | 94.53 | ⋆ | ⋆ | ⋆ | ⋆ | 97.07 | ⋆ | 97.51 | 96.74 |
| in629 | 107929.2600 | 40.10 | 95.76 | 94.64 | 97.16 | ⋆ | ⋆ | ⋆ | ⋆ | 98.12 | 98.29 |
| in630 | 105830.0620 | 54.00 | 94.55 | 93.68 | 99.65 | ⋆ | ⋆ | ⋆ | ⋆ | 99.75 | 99.75 |
| in631 | 116505.2440 | 31.18 | 94.57 | 94.02 | 96.91 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in632 | 104631.7140 | 52.88 | 92.98 | 90.76 | 95.18 | ⋆ | ⋆ | 99.59 | ⋆ | 98.28 | ⋆ |
| in633 | 105564.4000 | 65.72 | ⋆ | 90.90 | 98.72 | ⋆ | ⋆ | ⋆ | ⋆ | 94.20 | 94.02 |
| in634 | 108901.7300 | 47.86 | 94.31 | 91.85 | 93.34 | ⋆ | ⋆ | ⋆ | ⋆ | 93.80 | 93.80 |
| in635 | 112902.6340 | 39.75 | 92.44 | 86.62 | 92.44 | ⋆ | ⋆ | ⋆ | ⋆ | 94.72 | 94.72 |
| in636 | 106574.7480 | 48.82 | 92.64 | 91.03 | 98.23 | ⋆ | ⋆ | 99.07 | ⋆ | 97.76 | 93.89 |
| in637 | 107989.7280 | 33.07 | 92.70 | 91.77 | 99.01 | ⋆ | ⋆ | ⋆ | ⋆ | 99.01 | 99.01 |
| in638 | 112899.6320 | 30.01 | 97.48 | 88.95 | 92.88 | ⋆ | ⋆ | 97.48 | ⋆ | 97.48 | 97.48 |
| in639 | 108894.4550 | 43.77 | 92.99 | 94.68 | 95.35 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in640 | 108275.1328 | 53.59 | 96.04 | 91.48 | ⋆ | ⋆ | ⋆ | 99.08 | ⋆ | 96.60 | 96.60 |
| in641 | 109744.0625 | 56.06 | 98.77 | 92.27 | ⋆ | ⋆ | ⋆ | 99.73 | ⋆ | 98.77 | 98.77 |
| in642 | 114182.9688 | 40.41 | 91.12 | 90.13 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in643 | 104015.0240 | 13.04 | 94.62 | 91.75 | 97.97 | ⋆ | ⋆ | ⋆ | ⋆ | 97.97 | ⋆ |
| in644 | 108025.7490 | 60.10 | 98.22 | 93.57 | 98.34 | ⋆ | ⋆ | 99.03 | ⋆ | 99.03 | 98.25 |
| in645 | 105841.6720 | 37.99 | 92.97 | 90.05 | 96.51 | ⋆ | ⋆ | ⋆ | ⋆ | 96.08 | 97.25 |
| in646 | 107800.1030 | 33.71 | 93.84 | 94.90 | 95.98 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in647 | 107701.7109 | 51.25 | 90.90 | 93.95 | ⋆ | ⋆ | ⋆ | 95.92 | ⋆ | 95.37 | 95.37 |
| in648 | 105790.5900 | 37.28 | ⋆ | 96.66 | ⋆ | ⋆ | ⋆ | 99.93 | ⋆ | 99.55 | 99.55 |
| in649 | 107587.3710 | 40.30 | 88.79 | 94.52 | 95.70 | ⋆ | ⋆ | 98.63 | ⋆ | ⋆ | 99.79 |
| in650 | 103330.9010 | 45.80 | 92.36 | 93.79 | 96.86 | ⋆ | ⋆ | ⋆ | ⋆ | 97.02 | 97.50 |
| in651 | 103827.2970 | 55.97 | 95.17 | 94.21 | 98.85 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in652 | 107760.2480 | 28.48 | 94.24 | 97.48 | 97.48 | ⋆ | ⋆ | ⋆ | ⋆ | 97.20 | 96.19 |
| in653 | 113946.4766 | 34.60 | 91.38 | 89.41 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 94.22 | 94.22 |
| in654 | 111738.2310 | 35.91 | 94.25 | 89.29 | 98.26 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in655 | 111785.0640 | 44.33 | 91.74 | 88.65 | 97.39 | ⋆ | ⋆ | ⋆ | ⋆ | 96.13 | 96.13 |
| in656 | 112259.2750 | 43.93 | 90.25 | 96.67 | 96.67 | ⋆ | ⋆ | ⋆ | ⋆ | 94.21 | 95.51 |
| in657 | 112708.6560 | 37.47 | ⋆ | 93.86 | 96.88 | ⋆ | ⋆ | ⋆ | ⋆ | 95.65 | ⋆ |
| in658 | 110751.5340 | 38.17 | 91.70 | 91.29 | 93.87 | ⋆ | ⋆ | 96.80 | ⋆ | ⋆ | ⋆ |
| in659 | 106545.4270 | 39.45 | 94.76 | 96.16 | 96.16 | ⋆ | ⋆ | 99.03 | ⋆ | ⋆ | ⋆ |

Continue in next page...

Table 20: (continued).

| Inst. | Best | CORAL | CPLEX | $RG_{RK}$ | $BO_{MA}$ | $CA_{RA}$ | $CA_{LP}$ | $GA_{RA}$ | $GA_{LP}$ | $SD_{RA}$ | $SD_{LP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| in660 | 112293.6080 | 39.96 | 98.65 | 91.81 | 96.61 | ⋆ | ⋆ | 99.72 | ⋆ | 99.72 | 98.65 |
| in661 | 113106.6290 | 30.29 | 97.20 | 87.06 | 92.81 | ⋆ | ⋆ | ⋆ | ⋆ | 95.86 | 97.63 |
| in662 | 108298.0790 | 58.18 | 97.41 | 91.26 | 91.26 | ⋆ | ⋆ | ⋆ | ⋆ | 94.29 | 94.29 |
| in663 | 104826.7800 | 52.39 | 95.93 | 95.40 | 95.40 | ⋆ | ⋆ | ⋆ | ⋆ | 99.03 | 92.33 |
| in664 | 112866.8650 | 42.89 | 95.93 | 91.67 | 94.66 | ⋆ | ⋆ | 99.38 | ⋆ | 99.68 | 99.68 |
| in665 | 113002.6720 | 39.05 | 98.78 | 94.75 | 97.33 | ⋆ | ⋆ | 98.78 | ⋆ | 96.68 | 96.68 |
| in666 | 106441.1562 | 46.49 | ⋆ | 91.88 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.54 | 98.54 |
| in667 | 104683.7500 | 65.93 | 97.55 | 91.77 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 97.75 | 97.75 |
| in668 | 107483.1580 | 45.33 | 94.12 | 93.41 | 98.89 | ⋆ | ⋆ | ⋆ | ⋆ | 98.12 | 98.12 |
| in669 | 108163.4690 | 42.49 | 97.80 | 93.65 | 96.78 | ⋆ | ⋆ | ⋆ | ⋆ | 95.71 | 94.23 |
| in670 | 110200.8160 | 50.35 | 94.90 | 92.73 | 96.53 | ⋆ | ⋆ | ⋆ | ⋆ | 99.98 | 99.85 |
| in671 | 109306.8438 | 48.13 | 99.75 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 99.75 | 99.75 |
| in672 | 107534.8870 | 43.05 | 93.33 | 95.58 | 95.58 | ⋆ | ⋆ | ⋆ | ⋆ | 96.40 | 95.58 |
| in673 | 112320.2500 | 44.61 | 92.20 | 92.34 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.43 | 98.43 |
| in674 | 109558.2344 | 37.01 | 91.87 | 87.59 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 95.40 | 95.40 |
| in675 | 108131.9880 | 47.04 | ⋆ | 92.14 | 97.81 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in676 | 107052.1910 | 37.62 | 94.64 | 88.05 | 96.10 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in677 | 107831.5370 | 45.33 | 96.19 | 97.47 | 99.68 | ⋆ | ⋆ | 99.68 | ⋆ | 97.03 | 97.22 |
| in678 | 102422.8290 | 29.45 | ⋆ | 95.83 | 96.87 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in679 | 107982.4560 | 48.90 | 90.90 | 92.06 | 99.21 | ⋆ | ⋆ | 98.61 | ⋆ | 96.35 | 96.11 |
| in680 | 107500.5000 | 44.67 | 96.69 | 91.50 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 98.92 | 98.92 |
| in681 | 105237.2870 | 53.94 | 93.19 | 92.10 | 99.85 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ |
| in682 | 107948.1260 | 38.39 | 97.33 | 89.67 | 98.25 | ⋆ | ⋆ | ⋆ | ⋆ | 98.25 | 99.93 |
| in683 | 107777.6130 | 7.06 | 95.19 | 93.47 | 96.08 | ⋆ | ⋆ | ⋆ | ⋆ | 98.25 | 98.25 |
| in684 | 114153.7410 | 62.07 | 91.14 | 85.74 | 94.52 | ⋆ | ⋆ | ⋆ | ⋆ | 92.55 | 94.26 |
| in685 | 106686.6160 | 39.69 | 94.83 | 92.42 | 92.74 | ⋆ | ⋆ | 97.71 | ⋆ | 97.81 | 97.81 |
| in686 | 106364.3580 | 19.52 | 99.45 | 92.48 | 98.55 | ⋆ | 99.53 | 99.45 | 99.53 | 97.70 | 97.70 |
| in687 | 108301.4710 | 44.81 | 97.06 | 94.98 | 97.05 | ⋆ | ⋆ | ⋆ | ⋆ | 99.10 | 99.10 |
| in688 | 112012.5703 | 50.12 | 93.49 | 94.55 | ⋆ | ⋆ | ⋆ | 99.83 | ⋆ | 95.27 | 97.60 |
| in689 | 105968.1680 | 48.45 | 92.72 | 94.96 | 97.70 | ⋆ | ⋆ | ⋆ | ⋆ | 98.39 | 98.39 |
| in690 | 108489.7109 | 34.23 | 92.05 | 92.90 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 97.02 | 97.02 |
| in691 | 105564.6090 | 37.21 | 93.06 | 96.78 | 96.78 | ⋆ | ⋆ | ⋆ | ⋆ | 98.50 | 98.50 |
| in692 | 109226.0700 | 44.39 | 93.71 | 91.40 | 97.15 | ⋆ | ⋆ | ⋆ | ⋆ | 98.99 | 97.40 |
| in693 | 106719.6950 | 31.31 | 97.08 | 93.34 | 97.58 | ⋆ | ⋆ | 99.56 | ⋆ | 99.56 | 97.08 |
| in694 | 114477.0540 | 47.90 | 89.45 | 94.44 | 94.44 | ⋆ | ⋆ | 96.94 | ⋆ | 93.66 | 93.66 |
| in695 | 110240.9860 | 14.09 | 91.30 | 93.91 | 93.91 | ⋆ | ⋆ | 98.04 | ⋆ | ⋆ | ⋆ |
| in696 | 104559.9530 | 39.47 | ⋆ | 95.00 | 98.94 | ⋆ | ⋆ | ⋆ | ⋆ | 99.43 | 99.43 |
| in697 | 105958.6570 | 23.47 | 98.78 | 92.49 | 98.46 | ⋆ | ⋆ | ⋆ | ⋆ | 98.78 | 98.78 |
| in698 | 105463.0312 | 26.21 | 95.12 | 92.14 | ⋆ | ⋆ | ⋆ | ⋆ | ⋆ | 97.86 | 97.77 |
| in699 | 107132.3340 | 41.24 | 96.37 | 96.85 | 98.42 | ⋆ | ⋆ | 99.26 | ⋆ | 99.14 | 99.14 |
| in700 | 106730.6770 | 45.46 | 95.37 | 95.11 | 95.11 | ⋆ | ⋆ | ⋆ | ⋆ | 97.09 | 94.31 |

(Carlos E. Andrade) University of Campinas, Av. Albert Einstein, 1251, Campinas, SP, Brazil

*E-mail address*: andrade@ic.unicamp.br

(Rodrigo F. Toso) Department of Computer Science, Rutgers University, 110 Frelinghuysen Road, Piscataway, NJ 08854 USA

*E-mail address*: R.F. Toso

(Mauricio G.C. Resende) AT&T Labs Research, 200 South Laurel Avenue, Room A5-1F34, Middletown, NJ 07748 USA

*E-mail address*: mgcr@research.att.com

(Flávio K. Mizasawa) University of Campinas, Av. Albert Einstein, 1251, Campinas, SP, Brazil

*E-mail address*: fkm@ic.unicamp.br