PROBABILITY DISTRIBUTION OF SOLUTION TIME IN GRASP: AN EXPERIMENTAL INVESTIGATION

RENATA M. AIEX, MAURICIO G.C. RESENDE, AND CELSO C. RIBEIRO

Abstract. A GRASP (greedy randomized adaptive search procedure) is a multi-start metaheuristic for combinatorial optimization. We study the probability distributions of solution time to a sub-optimal target value in five GRASPs that have appeared in the literature and for which source code is available. The distributions are estimated by running 12,000 independent runs of the heuristic. Standard methodology for graphical analysis is used to compare the empirical and theoretical distributions and estimate the parameters of the distributions. We conclude that the solution time to a sub-optimal target value fits a two-parameter exponential distribution. Hence, it is possible to approximately achieve linear speed-up by implementing GRASP in parallel.

1. Introduction

A greedy randomized adaptive search procedure (GRASP) [8, 9, 11] is a multi-start or iterative process, in which each GRASP iteration consists of two phases. In a construction phase, a feasible solution is produced and in a local search phase, a local optimum in the neighborhood of the constructed solution is sought. The best overall solution is kept as the result.

In the construction phase, a feasible solution is iteratively constructed, one element at a time. The basic GRASP construction phase is similar to the semi-greedy heuristic proposed independently by Hart and Shogan [14]. At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements (i.e. those that can be added to the solution) in a candidate list C with respect to a greedy function $g : C \to \mathbb{R}$. This function measures the (myopic) benefit of selecting each element. The heuristic is adaptive because the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous element. The probabilistic component of a GRASP is characterized by randomly choosing one of the best candidates in the list, but not necessarily the top candidate. The list of best candidates is called the *restricted candidate list* (RCL).

It is almost always beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the neighborhood.

Date: December 2000.

AT&T Labs Research Technical Report: 00.7.2. Published in *J. of Heuristics*, vol. 8, pp. 343–373, 2002.

TABLE 1. CPU time (in seconds) and speed-up on MAX-SAT problems. Average speed-up is shown for 5, 10, and 15 processors.

	1 processor	5 processors		10 processors		15 processors	
problem	time	time	speed-up	time	speed-up	time	speed-up
jnh201	310.4	62.8	4.9	30.5	10.2	22.2	14.0
jnh202	312.2	59.8	5.2	31.2	10.0	23.4	13.3
jnh203	351.2	72.3	4.9	35.2	10.0	23.2	15.1
jnh205	327.8	63.4	5.2	32.1	10.2	22.5	14.6
jnh207	304.7	56.7	5.4	29.6	10.3	19.8	15.4
jnh208	355.2	65.6	5.4	33.2	10.7	21.0	16.9
jnh209	339.0	60.5	5.6	33.6	10.1	21.6	15.7
jnh210	318.5	57.6	5.5	32.5	9.8	20.8	15.3
jnh301	414.5	85.3	4.9	45.2	9.2	28.3	14.6
jnh302	398.7	88.6	4.5	48.2	8.3	27.0	14.7
average speed-up:			5.2		9.9		15.0

As with any multi-start heuristic for combinatorial optimization, a GRASP can be implemented in parallel by dividing the k independent iterations among ρ processors. Another approach is to give a target value τ of the objective function to each processor and execute the algorithm until the first processor finds a solution with value at least as good as τ, at which point all processors halt. Some care is needed to assure that no two iterations start with identical random number generator seeds [22]. These are examples of *multiple independent walk* parallelism [36].

Many parallel implementations of GRASP using the above strategies have been reported in the literature, e.g. [17, 18, 19, 21, 22]. In most of these papers, a common observation was made. The speedups in the measured running times were proportional to the number of processors. A typical example can be seen in Pardalos, Pitsoulis, and Resende [22] where, for a PVM-based implementation of a parallel GRASP for the MAX-SAT, average speed-ups almost identical to the number of processors were measured (see Table 1).

This observation can be explained if the random variable *solution time to target* is exponentially distributed, as indicated by the following proposition [36].

Proposition 1. *Let $P_\rho(t)$ be the probability of not having found a given (target) solution in t time units with ρ independent processes. If $P_1(t) = e^{-t/\lambda}$ with $\lambda \in \mathbb{R}^+$, i.e. P_1 corresponds to an exponential distribution, then $P_\rho(t) = e^{-\rho t/\lambda}$.*

The above proposition follows from the definition of the exponential distribution. It implies that the probability of finding a solution of a given value in time ρt with a sequential process is equal to the probability of finding a solution at least as good as that given value in time t with ρ independent parallel processes. Hence, it is possible to achieve linear speed-up in solution time to target solution by multiple independent processes.

An analogous proposition can be stated for a two parameter (shifted) exponential distribution.

Proposition 2. *Let $P_\rho(t)$ be the probability of not having found a given (target) solution in t time units with ρ independent processes. If $P_1(t) = e^{-(t-\mu)/\lambda}$ with $\lambda \in \mathbb{R}^+$ and $\mu \in \mathbb{R}$, i.e. P_1 corresponds to a two parameter exponential distribution, then $P_\rho(t) = e^{-\rho(t-\mu)/\lambda}$.*

Analogously, this proposition follows from the definition of the two parameter exponential distribution. It implies that the probability of finding a solution of a given value in time ρt with a sequential process is equal to $1 - e^{-(\rho t-\mu)/\lambda}$ while the probability of finding a solution at least as good as that given value in time t with ρ independent parallel processes is $1 - e^{-\rho(t-\mu)/\lambda}$. Note that if $\mu = 0$, then both probabilities are equal and correspond to the non-shifted exponential distribution. Furthermore, if $\rho\mu \ll \lambda$, then the two probabilities are approximately equal and it is possible to approximately achieve linear speed-up in solution time to target solution by multiple independent processes.

This behavior has been noted in a number of metaheuristics. These include simulated annealing [6, 20]; iterated local search algorithms for the traveling salesman problem [7], where it is shown that the probability of finding a sub-optimal solution is given by a shifted exponential distribution, allowing for the time to find the first local optimum; tabu search, provided that the search starts from a local optimum [1, 35]; and WalkSAT [34] on hard random 3-SAT problems [15].

The objective of this paper is to determine if the solution times for GRASP also have this property, i.e., they fit a two parameter exponential distribution. To do this, we consider five GRASPs that have been reported in the literature and for which we have source code:

(1) maximum independent set [10, 28];
(2) quadratic assignment problem [16, 29];
(3) graph planarization [32, 33];
(4) maximum weighted satisfiability [30, 31];
(5) maximum covering [26].

For each GRASP, we selected four test problems from the literature. For each of these instances, we determined three solution target values spread out between minimum and maximum values produced by GRASP. For each target value, we measured running times to find a solution at least as good as the target and studied these distributions.

The remainder of this paper is organized as follows. In Section 2, we give a brief overview of each of the five GRASPs used in this study. The experimental design is described in Section 3. The experimental results are reported in Section 4. In Section 5, we make concluding remarks.

2. Five GRASP implementations

In this section, we briefly describe the five GRASPs used in the experiments. For each GRASP, we define the combinatorial optimization problem it solves, the construction phase and the local search phase.

2.1. GRASP for maximum independent set.

A GRASP for maximum independent set was introduced by Feo and Resende [10]. Fortran subroutines that implement this GRASP are found in Resende, Feo, and Smith [28].

2.1.1. *Problem definition.* Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. Vertices $u, v \in V$ are *nonadjacent* if $(u, v) \notin E$. A subset of the vertices $S \subseteq V$ is *independent* if all vertices in S are pairwise nonadjacent. In the *maximum independent set problem* one wants to find an independent set having the largest cardinality.

2.1.2. *Construction phase.* The algorithm initializes a working graph $\tilde{G} = (\tilde{V}, \tilde{E})$ to be the original graph, and sets the independent set S empty. The independent set is built up one vertex at a time. The greedy function that guides the construction is vertex degree with respect to the working graph. It selects among the working graph vertices, the one with minimum degree and places that vertex in the independent set. The greedy function is adaptive, since it changes with the selection of each independent vertex.

Let $\underline{d}$ and $\overline{d}$ be, respectively, the minimum and maximum degrees over all working vertices, i.e.

$$\underline{d} = \min_{v \in \tilde{V}} \{d(v, \tilde{G})\} \text{ and } \overline{d} = \max_{v \in \tilde{V}} \{d(v, \tilde{G})\},$$

where $d(v, \tilde{G})$ is the degree of vertex v with respect to $\tilde{G}$. The *restricted candidate list* (RCL) is the set of vertices

$$\text{RCL} = \{v \in \tilde{V} \mid d(v, \tilde{G}) \leq \underline{d} + \alpha(\overline{d} - \underline{d})\},$$

where the parameter α controls the size of the RCL and is such that $0 \leq \alpha \leq 1$. The vertex selection in the GRASP construction phase is random, restricted to vertices in the RCL.

2.1.3. *Local search phase.* A $(2, 1)$-exchange local search heuristic for the maximum independent set problem seeks a larger independent set by removing a single vertex x from the independent set S and replacing it by two nonadjacent vertices u and v, such that u and v are not adjacent to any vertex in $S \setminus \{x\}$. If such an exchange is found, the procedure is recursively applied on the new larger independent set. A locally optimal solution is detected when no further exchange is possible.

2.2. **GRASP for quadratic assignment.** Li, Pardalos, and Resende [16] introduce a GRASP for the quadratic assignment problem (QAP) and describe Fortran subroutines for this GRASP in [29]. A specialization of this GRASP for sparse QAPs together with Fortran subroutines are presented in [23]. A parallel version of this GRASP can be found in [21]. Improvements to the construction and local search phases are described in [12, 24, 25].

2.2.1. *Problem definition.* Given a set $\mathcal{N} = \{1, 2, \ldots, n\}$ and $n \times n$ matrices $F = (f_{ij})$ and $D = (d_{kl})$, the quadratic assignment problem (QAP) can be stated as follows:

$$\min_{p \in \Pi_{\mathcal{N}}} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{p(i)p(j)},$$

where $\Pi_{\mathcal{N}}$ is the set of all permutations of $\mathcal{N}$.

2.2.2. *Construction phase.* The construction phase has two stages. In stage 1, two assignments are produced, i.e. facility i is assigned to site k and facility j is assigned to site l. The idea is to assign facilities with high interaction (having high f_{ij} values) to nearby sites (site pairs with low d_{kl} values). To do this, the procedure sorts inter-site distances in increasing order and inter-facility flows in decreasing order. Let $d_{k_1,l_1} \leq d_{k_2,l_2} \leq \cdots \leq d_{k_p,l_p}$ and $f_{i_1,j_1} \geq f_{i_2,j_2} \geq \cdots \geq f_{i_p,j_p}$ be the sorted values, where $p = n^2 - n$. The products $d_{k_1,l_1} \cdot f_{i_1,j_1}, d_{k_2,l_2} \cdot f_{i_2,j_2}, \ldots, d_{k_p,l_p} \cdot f_{i_p,j_p}$ are then sorted in increasing order. Among the smallest $d_{kl} \cdot f_{ij}$ products, one (corresponding to the pair of stage 1 assignments) is selected at random. Sorting all of the $p = n^2 - n$ distances and flows is inefficient and offers little benefit. Instead, only the best $n_\beta = \beta p$ values are sorted, where β is a parameter such that $0 < \beta \leq 1$. Among these n_β pairs of assignments, a pair is selected at random from the set of αn_β assignments having the smallest $d_{kl} \cdot f_{ij}$ products, where α is such that $0 < \alpha \leq 1$.

In stage 2 of the construction phase, the remaining $n-2$ facility-site assignments are made sequentially. The idea is to favor assignments that have small interaction cost with the set of previously-made assignments. Let Γ be the set of q assignments at a given point in the construction phase, i.e. $\Gamma = \{(i_1, k_1), (i_2, k_2), \ldots, (i_q, k_q)\}$. The cost of assigning facility j to site l, with respect to the already-made assignments, is defined to be

$$c_{jl} = \sum_{(i,k)\in\Gamma} f_{ij} d_{kl}.$$

All costs of unassigned facility-site pairs (j, l) are sorted in increasing order. Of the pairs having the least $\alpha \cdot |\Gamma|$ costs, one is selected at random and is added to the set Γ. The procedure is repeated until $n-1$ assignments are made. The remaining facility is then assigned to the remaining site.

2.2.3. *Local search phase.* In the local search phase of this GRASP, a 2-exchange neighborhood search is conducted on the constructed solution. There, all possible 2-swaps of facility-locations are considered. If a swap improves the cost of the assignment, it is accepted. The procedure continues until no swap improves the solution value.

2.3. **GRASP for graph planarization.** A GRASP for graph planarization was introduced in Resende and Ribeiro [32]. Fortran subroutines for their algorithm are described in [33].

2.3.1. *Problem definition.* A graph is said to be *planar* if it can be drawn on the plane in such a way that no two of its edges cross. Given a graph $G = (V, E)$ with vertex set V and edge set E, the objective of *graph planarization* is to find a minimum cardinality subset of edges $F \subseteq E$ such that the graph $G' = (V, E \setminus F)$, resulting from the removal of the edges in F from G, is planar. This problem is also known as the *maximum planar subgraph* problem.

2.3.2. *Two phase heuristic.* The GRASP described in this section is based on the separation of the computation into two phases [13]. The first phase consists in devising a linear permutation of the nodes of the input graph, followed by placing them along a line. The second phase determines two sets of edges that may be represented without crossings above and below that line, respectively.

The GRASP construction and local search are applied to the first phase, where a linear permutation of the nodes is determined.

2.3.3. *Construction phase.* After the first k nodes of the permutation have been determined, say $v_1, v_2, \cdots, v_k$, the next node v_{k+1} is selected at random from the nodes adjacent to v_k in G having the lowest degrees in the subgraph G_k of G induced by $V \setminus \{v_1, v_2, \cdots, v_k\}$. If there is no node of G_k adjacent to v_k in G, then v_{k+1} is selected at random from a set of low degree nodes in G_k.

2.3.4. *Local search phase.* The local search phase explores the neighborhood of the current permutation by swapping the positions of two nodes at a time, attempting to reduce the number of possible edge crossings.

2.3.5. *Post-optimization.* Each iteration of this GRASP produces three edge sets: $\mathcal{B}$ (blue edges, which are drawn above the line), $\mathcal{R}$ (red edges, which are drawn below the line), and $\mathcal{P}$ (the remaining edges, which are referred to as the *pale* edges). By construction, $\mathcal{B}$, $\mathcal{R}$, and $\mathcal{P}$ are such that no red or pale edge can be colored blue. Likewise, pale edges cannot be colored red. However, if there exists a pale edge $p \in \mathcal{P}$ such that all blue edges that cross with p (let $\hat{\mathcal{B}}_p \subseteq \mathcal{B}$ be the set of those blue edges) do not cross with any red edge $r \in \mathcal{R}$, then all blue edges $b \in \hat{\mathcal{B}}_p$ can be colored red and p can be colored blue. In case this reassignment of colors is possible, then the size of the planar subgraph is increased by one edge. This post-optimization procedure is incorporated at the end of each GRASP iteration.

2.4. **GRASP for MAX-SAT.** A GRASP for satisfiability was first proposed in Resende and Feo [27]. This GRASP was generalized to handle MAX-SAT problems by Resende, Pitsoulis, and Pardalos [30]. A parallel version of this algorithm is described in [22] and Fortran subroutines are presented in [31].

2.4.1. *Problem definition.* Let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$ be m clauses, involving n Boolean variables x_1, x_2, ..., x_n, which can take on only the values `true` or `false` (1 or 0). In addition, for each clause $\mathcal{C}_i$, there is an associated nonnegative weight w_i. Define clause i to be

$$\mathcal{C}_i = \bigvee_{j=1}^{n_i} l_{ij},$$

where n_i is the number of literals in clause $\mathcal{C}_i$, and literal $l_{ij} \in \{x_i, \bar{x}_i \mid i = 1, \ldots, n\}$. A clause is said to be *satisfied* if it evaluates to `true`. In the weighted *Maximum Satisfiability Problem* (MAX-SAT), one is to determine the assignment of truth values to the n variables that maximizes the sum of the weights of the satisfied clauses.

2.4.2. *Construction phase.* A feasible solution to a MAX-SAT instance is described by $x \in \{0,1\}^n$. Let $w(x)$ is the sum of the weights of the clauses satisfied by x. The construction phase solution is built, one element at a time, guided by a greedy function and randomization. Since in the MAX-SAT problem there are n variables to be assigned, each construction phase consists of n iterations.

The idea behind the greedy function is to maximize the total weight of yet-unsatisfied clauses that become satisfied after the assignment of each construction phase iteration. For $i \in N$, let Γ_i^+ be the set of unassigned clauses that would become satisfied if variable x_i were to be set to `true`. Likewise, let Γ_i^- be the set

of unassigned clauses that would become satisfied if variable x_i were to be set to `false`. Define

$$\gamma_i^+ = \sum_{j \in \Gamma_i^+} w_j \text{ and } \gamma_i^- = \sum_{j \in \Gamma_i^-} w_j.$$

The greedy choice is to select the variable x_k with the largest γ_k^+ or γ_k^- value and set it to the corresponding truth value. If $\gamma_k^+ > \gamma_k^-$, then the assignment $x_k = 1$ is made, else $x_k = 0$. Note that with every assignment made, the sets Γ_i^+ and Γ_i^- change for all i such that x_i is not assigned a truth value, to reflect the new assignment. This consequently changes the values of γ_i^+ and γ_i^-, characterizing the adaptive component of the heuristic.

Let

$$\gamma^* = \max\{\gamma_i^+, \gamma_i^- \mid x_i \text{ yet unassigned}\}$$

and

$$\gamma_* = \min\{\gamma_i^+, \gamma_i^- \mid x_i \text{ yet unassigned}\},$$

and let α $(0 \leq \alpha \leq 1)$ be the restricted candidate parameter. A new value for α is selected, at random, at each iteration, from the uniform distribution $U[0, 1]$. A candidate $x_i =$ `true` is inserted into the RCL if $\gamma_i^+ \geq \gamma_* + \alpha \cdot (\gamma^* - \gamma_*)$. Likewise, a candidate $x_i =$ `false` is inserted if $\gamma_i^- \geq \gamma_* + \alpha \cdot (\gamma^* - \gamma_*)$.

2.4.3. *Local search phase.* To define the local search procedure, some preliminary definitions have to be made. Given a truth assignment $x \in \{0, 1\}^n$, define the *1-flip neighborhood* $N(x)$ to be the set of all vectors $y \in \{0, 1\}^n$ such that $\| x - y \|_2 = 1$. If x is interpreted as a vertex of the n-dimensional unit hypercube, then its neighborhood consists of the n vertices adjacent to x. If we denote by $w(x)$ the total weight of the clauses satisfied by the truth assignment x, then the truth assignment x is a *local maximum* if and only if $w(x) \geq w(y)$, for all $y \in N(x)$. Starting with a truth assignment x, the local search finds the local maximum y in $N(x)$. If $y \neq x$, it sets $x = y$. This process is repeated until no further improvement is possible.

2.5. **GRASP for maximum covering.** A GRASP for the maximum covering problem is described in Resende [26].

2.5.1. *Problem definition.* The maximum covering problem can be stated as: Let $J = \{1, 2, \ldots, n\}$ denote the set of n potential facility locations. Define n finite sets $P_1, P_2, \ldots, P_n$, each corresponding to a potential facility location, such that $I = \cup_{j \in J} P_j = \{1, 2, \ldots, m\}$ is the set of the m demand points that can be covered by the n potential facilities. With each demand point $i \in I$, we associate a weight $w_i \geq 0$. A *cover* $J^* \subseteq J$ covers the demand points in set $I^* = \cup_{j \in J^*} P_j$ and has an associated weight $w(J^*) = \sum_{i \in I^*} w_i$. Given the number $p > 0$ of facilities to be placed, we wish to find the set $J^* \subseteq J$ that maximizes $w(J^*)$, subject to the constraint that $|J^*| = p$.

2.5.2. *Construction phase.* Since in the maximum covering problem there are p facility locations to be chosen, each construction phase consists of p iterations, with one location chosen per iteration.

To define a restricted candidate list, we rank the yet unchosen facility locations according to an adaptive greedy function. Let J^* denote the set (initially empty) of chosen facility locations being built in the construction phase. At any construction

phase iteration, let Γ_j be the set of additional uncovered demand points that would become covered if facility location j (for $j \in J \setminus J^*$) were to be added to J^*. Define the *greedy function*

$$\gamma_j = \sum_{i \in \Gamma_j} w_i$$

to be the incremental weight covered by the choice of facility location $j \in J \setminus J^*$. The greedy choice is to select the facility location k having the largest γ_k value. Note that with every selection made, the sets Γ_j, for all yet unchosen facility location indices $j \in J \setminus J^*$, change to reflect the new selection. This consequently changes the values of the greedy function γ_j, characterizing the adaptive component of the heuristic.

Let

$$\gamma^* = \max\{\gamma_j \mid \text{facility location } j \text{ is still unselected, i.e. } j \in J \setminus J^*\}$$

and α be the restricted candidate parameter ($0 \leq \alpha \leq 1$). We say a facility location j is a *potential candidate*, and is added to the RCL, if $\gamma_j \geq \alpha \times \gamma^*$. A location is selected at random from the RCL and is added to the solution.

2.5.3. *Local search phase.* Two solutions (sets of facility locations) J^1 and J^2 are said to be neighbors in the 2-exchange neighborhood if they differ by exactly one element, i.e. $\mid J^1 \cap \Delta J \mid \; = \; \mid J^2 \cap \Delta J \mid \; = 1$, where $\Delta J = (J^1 \cup J^2) \setminus (J^1 \cap J^2)$. The local search starts with a set J^* of p facility locations, and at each iteration attempts to find a pair of locations $s \in J^*$ and $t \in J \setminus J^*$ such that $w(J^* \setminus \{s\} \cup \{t\}) > w(J^*)$. If such a pair exists, then location s is replaced by location t in J^*. A solution is locally optimal with respect to this neighborhood if there exists no pairwise exchange that increases the total weight of J^*.

3. Experimental design

In this section we describe the experimental design. We analyze five GRASPs that have appeared in the literature and for which source code is available. For each of these algorithms, we select four test problems to study the probability distribution of solution time. The hypothesis of this paper is that CPU times fit a two parameter exponential distribution. We measure the CPU time to find an objective function value at least as good as a given target value. This is done for three different target values for each test problem. These values are spread out between a value far from the optimal and the best value produced by GRASP. Each GRASP is run $n = 200$ times for all instance/target combinations. For each of the 200 runs of each combination, the random number generator is initialized with a distinct seed and therefore the runs are independent. To compare the empirical and the theoretical distributions, we follow a standard graphical methodology for data analysis [4]. In the remainder of this section we describe this methodology.

For each instance/target pair, the running times are sorted in increasing order. We associate with the i-th sorted running time (t_i) a probability $p_i = (i - \frac{1}{2})/n$, and plot the points $z_i = (t_i, p_i)$, for $i = 1, \ldots, n$. We comment on this choice of p_i later in this section. Figure 1 illustrates this cumulative probability distribution plot for one of the instance/target pairs.

To estimate the parameters of the two-parameter exponential distribution, we first draw the theoretical quantile-quantile plot (or Q-Q plot) for the data. To

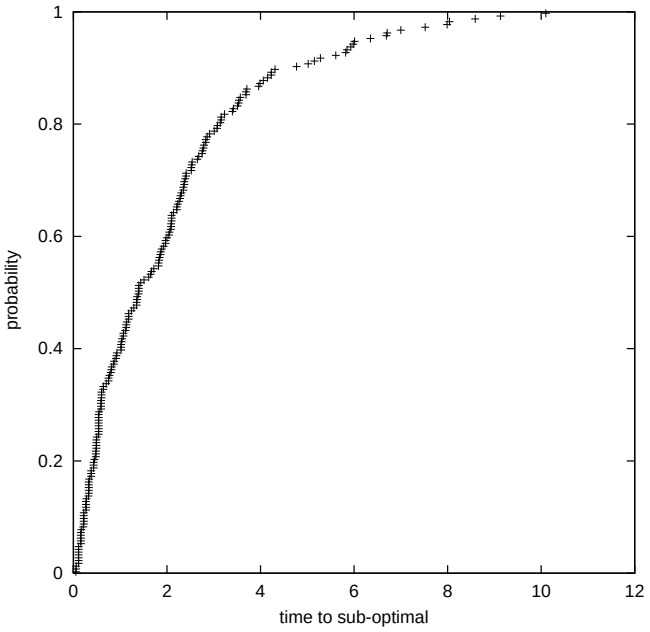

FIGURE 1. Cumulative probability distribution plot of measured data.

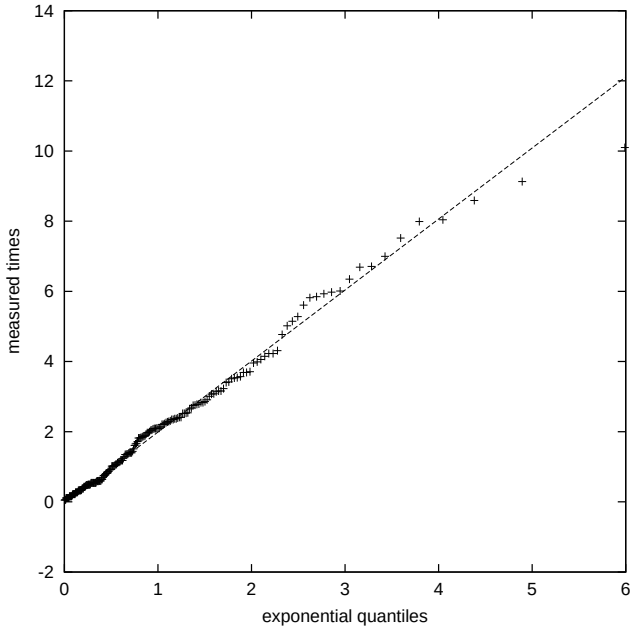

FIGURE 2. Q-Q plot showing fitted line.

describe Q-Q plots, we recall that the cumulative distribution function for the two-parameter exponential distribution is given by

$$F(t) = 1 - e^{-(t-\mu)/\lambda},$$

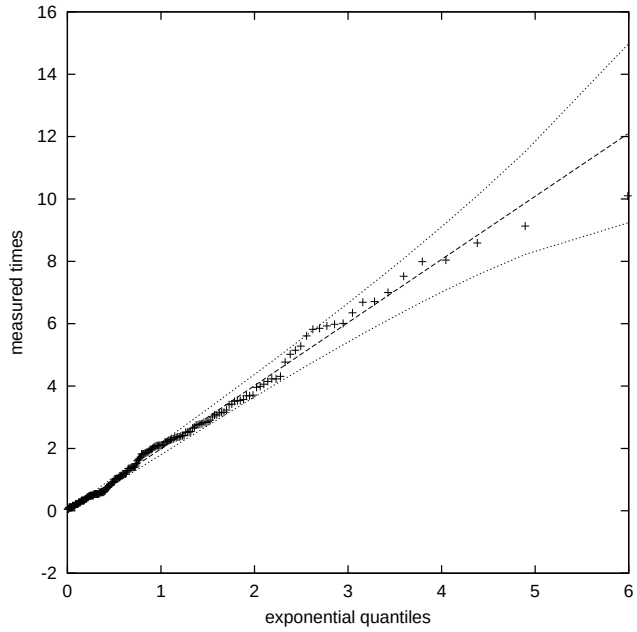

FIGURE 3. Q-Q plot with variability information.

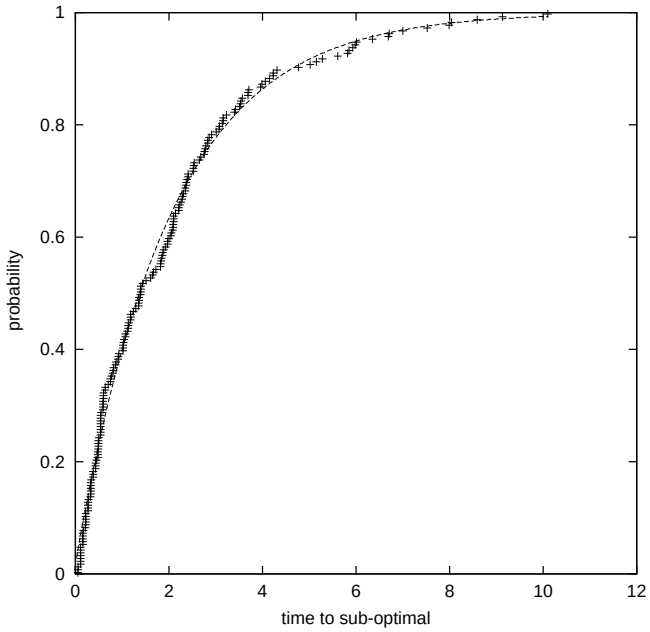

FIGURE 4. Superimposed empirical and theoretical distributions.

where λ is the mean of the distribution data (and indicates the spread of the data) and μ is the shift of the distribution with respect to the ordinate axis.

For each value p_i, $i = 1, \ldots, n$, we associate a p_i-quantile $Qt(p_i)$ of the theoretical distribution. For each p_i-quantile we have, by definition, that

$$F((Qt(p_i)) = p_i.$$

Hence, $Qt(p_i) = F^{-1}(p_i)$ and therefore, for the two-parameter exponential distribution, we have

$$Qt(p_i) = -\lambda \ln(1 - p_i) + \mu.$$

The quantiles of the data of an empirical distribution are simply the (sorted) raw data. Note that if we were to use $p_i = i/n$, for $i = 1, \ldots, n$, then $Qt(p_n)$ would be undefined.

A theoretical quantile-quantile plot (or theoretical Q-Q plot) is obtained by plotting the quantiles of the data of an empirical distribution against the quantiles of a theoretical distribution. This involves three steps. First, the data (in our case, the measured times) are sorted in ascending order. Second, the quantiles of the theoretical exponential distribution are obtained. Finally, a plot of the data against the theoretical quantiles is made.

In a situation where the theoretical distribution is a close approximation of the empirical distribution, the points in the Q-Q plot will have a nearly straight configuration. If the parameters λ and μ of the theoretical distribution that best fits the measured data could be estimated a priori, the points in a Q-Q plot would tend to follow the line $x = y$. Alternatively, in a plot of the data against a two-parameter exponential distribution with $\lambda = 1$ and $\mu = 0$, the points would tend to follow the line $y = \hat{\lambda}x + \hat{\mu}$. This means that a single theoretical Q-Q plot compares a set of data not just to one theoretical distribution, but simultaneously to a whole family of distributions. Consequently, parameters λ and μ of the two-parameter exponential distribution can be estimated, respectively, by the slope $\hat{\lambda}$ and intercept $\hat{\mu}$ of the line depicted in the Q-Q plot.

The Q-Q plot shown in Figure 2 is obtained by plotting the measured times in the ordinate against the quantiles of a two-parameter exponential distribution with $\lambda = 1$ and $\mu = 0$ in the abscissa, given by $-\ln(1 - p_i)$ for $i = 1, \ldots, n$. To avoid possible distortions caused by outliers, we do not estimate the distribution mean with the data mean or by linear regression on the points of the Q-Q plot. Instead, we estimate the slope $\hat{\lambda}$ of line $y = \lambda x + \mu$ using the upper quartile q_u and lower quartile q_l of the data. The upper and lower quartiles are, respectively, the $Q(\frac{1}{4})$ and $Q(\frac{3}{4})$ quantiles, respectively. We take

$$\hat{\lambda} = (z_u - z_l)/(q_u - q_l)$$

as an estimate of the slope, where z_u and z_l are the u-th and l-th points of the ordered measured times, respectively. This informal estimation of the distribution of the measured data mean is robust since it will not be distorted by a few outliers [4].

To analyze the straightness of the Q-Q plots, we superimpose them with variability information. For each plotted point, we show plus and minus one standard deviation in the vertical direction from the line fitted to the plot. An estimate of the standard deviation for point z_i, $i = 1, \ldots, n$, of the Q-Q plot is

$$\hat{\sigma} = \hat{\lambda}\sqrt{\frac{p_i}{(1 - p_i)n}}.$$

Figure 3 shows an example of a Q-Q plot with superimposed variability information.

When observing a theoretical quantile-quantile plot with superimposed standard deviation information, one should avoid turning such information into a formal test. One important fact that must be kept in mind is that the natural variability of the data generates departures from the straightness, even if the model of the distribution is valid. The most important reason for portraying standard deviation is that it gives us a sense of the relative variability of the points in the different regions of the plot. However, since one is trying to make simultaneous inferences from many individual inferences, it is difficult to use standard deviations to judge departures from the reference distribution. For example, the probability that a particular point deviates from the reference line by more than two standard deviations is small. But the probability that at least one of the data points deviates from the line by two standard deviations is probably much greater. In order statistics, this is made more difficult by the high correlation that exists between neighboring points. If one plotted point deviates by more than one standard deviation, there is a good chance that a whole bunch of them will too. Another point to keep in mind is that standard deviations vary substantially in the Q-Q plot, as can be observed in the Q-Q plot in Figure 3 that the standard deviations of the points near the high end are substantially larger then the standard deviation of the other end.

Once the two parameters of the distribution are estimated, a superimposed plot of the empirical and theoretical distributions can be made. Figure 4 shows this plot corresponding to the Q-Q plot in Figure 2.

4. Computational results

In this section, we present the computational results. We describe the computer environment used to conduct the experiments, the instances selected for each of the five GRASPs, and present for each GRASP/instance/target triplet its Q-Q plot with variability information, the two estimated parameters, and superimposed plots of the empirical and theoretical distributions.

4.1. **Computer environment.**

The experiments were done on an SGI Challenge computer (28 196 MHz MIPS R10000 processors) with 7.6 Gb of memory. Each run used a single processor.

The algorithms were coded in Fortran and were modified minimally to produce the data needed for the experiments. CPU times were measured with the system function `etime`. The codes were compiled with the SGI MIPSpro F77 compiler. The GRASPs for maximum independent set and quadratic assignment were compiled using flags `-Ofast -u` and the GRASP for maximum satisfiability was compiled using flags `-Ofast -static`. The GRASP for maximum covering was compiled using flags `-O3 -r4 -64` and the GRASP for planarization was compiled using flags `-O3 -static`.

4.2. **Test problems.**

The test problem names, their best known solutions, and respective target values are shown in Tables 2– 6.

The four problems used to study the GRASP for maximum independent set were chosen from a much studied class of random graphs [2], denoted by $G_{n,p}$. Such graphs have n nodes and each edge (i, j), $i, j = 1, \ldots, n, \quad i \neq j$, exists with probability p. The experiment consisted in running the algorithm on four random instances of $G_{n,p}$ $(n = 1000, p = 0.5)$ generated with the Fortran code in the distribution [28].

TABLE 2. Maximum independent set test problems with best known solutions (bks), target values, and parameter estimates.

			estimates	
problem	bks	target	$\hat{\mu}$	$\hat{\lambda}$
270002	15	13	0.015	0.146
		14	-0.018	1.612
		15	25.560	291.715
270003	15	13	0.068	0.147
		14	0.142	5.797
		15	1.849	223.491
270004	15	13	0.034	0.092
		14	-0.035	2.024
		15	1.339	30.248
270006	15	13	0.021	0.137
		14	-0.013	1.383
		15	38.909	516.965

TABLE 3. Quadratic assignment problem test problems with best known solutions, target values, and parameter estimates.

			estimates	
problem	bks	target	$\hat{\mu}$	$\hat{\lambda}$
chr25a	3796	5023	0.016	0.221
		4721	0.012	1.147
		4418	0.460	8.401
kra30b	91420	94675	0.021	0.076
		93590	0.024	0.261
		92505	-0.041	2.045
sko42	15812	16389	0.051	0.049
		16222	0.044	0.244
		16055	0.173	1.955
tho40	240516	247160	0.105	0.419
		245396	0.255	2.163
		243632	3.397	19.413

For the GRASP for QAP, test problems chr25a, kra30b, sko42, and tho40, were chosen from the suite of QAP test programs QAPLIB [3]. The problems are pure quadratic assignment problems that have at least one symmetric distance or flow matrix. Their dimensions (n) range from 25 to 42.

The GRASP for graph planarization was tested for four problems (g17, tg100.10, rg100.1, and rg150.1) chosen from a set of 75 test problems described in the literature [5, 13]. The dimensions of the selected problems range from 100 to 150 vertices and 261 to 742 edges.

The test problems for the MAX-SAT problem were chosen among the instances reported in Resende, Pitsoulis, and Pardalos [30]. Problems jnh11 and jnh12 have 100 variables and 800 clauses, problem jnh212 has 100 variables and 850 clauses, and problem jnh306 has 100 variables and 900 clauses.

TABLE 4. Graph planarization test problems with best known solutions, target values, and parameter estimates.

			estimates	
problem	bks	target	$\hat{\mu}$	$\hat{\lambda}$
g17	236	222	2.564	8.723
		227	-0.738	72.498
		231	-19.991	763.081
tg100.10	277	215	0.575	0.690
		226	0.747	5.120
		236	-10.540	181.038
rg100.1	162	154	0.135	0.563
		157	0.062	3.983
		159	-0.148	25.954
rg150.1	231	215	0.531	0.722
		220	0.368	6.961
		225	3.495	286.668

TABLE 5. Maximum satisfiability test problems with best known solutions, target values, and parameter estimates.

maximum satisfiability				
			estimates	
problem	bks	target	$\hat{\mu}$	$\hat{\lambda}$
jnh11	420753	418851	0.061	0.136
		419485	0.063	0.876
		420119	0.860	24.903
jnh12	420925	417664	0.053	0.046
		418751	0.044	0.233
		419838	0.064	2.797
jnh212	394238	393145	0.033	0.707
		393506	-0.226	4.148
		393866	-0.261	46.058
jnh306	444838	441720	0.059	0.058
		442759	0.062	0.219
		443798	-0.198	3.509

The test problems for the GRASP for maximum covering (r24-500, r25-250, r54-100, and r55-100) were generated randomly using the generator described in Resende [26]. All instances have 1000 potential location sites and demand points varying from 7425 to 9996. The number of facilities to be located varies from 100 to 500.

4.3. **Generating the data points.** 200 independent runs of each GRASP were done for each instance/target pair. In each run, the GRASP was halted after a solution at least as good as the target was found and the total CPU time (in seconds) was recorded. With the 200 data points generated, a Q-Q plot with

TABLE 6. Maximum covering test problems with best known solutions, target values, and parameter estimates.

maximum covering				
			estimates	
problem	bks	target	$\hat{\mu}$	$\hat{\lambda}$
r24-500	33343542	33330441	-2.257	109.827
		33334808	11.960	229.850
		33339175	2.273	669.501
r25-250	20606926	20567005	1.042	3.319
		20580312	0.716	14.555
		20593619	4.279	101.40
r54-100	39684669	39332719	6.272	42.187
		39450036	17.803	272.29
		39567352	73.320	3978.427
r55-100	39504338	39037219	6.917	9.063
		39192925	3.190	37.672
		39348631	-10.888	279.470

variability information and a superimposed plot of the empirical and theoretical distributions were made.

4.4. **Q-Q plots and theoretical distributions.** In this subsection, we present Q-Q plots with variability information with the corresponding plots of superimposed empirical and theoretical distributions.

Each Q-Q plot figure is composed of 12 Q-Q plots, one for each instance/target pair. Likewise, each superimposed empirical and theoretical distribution figure has 12 plots, one for each instance/target pair. Each plot is made up of 200 data points. Each figure has four rows of plots, each corresponding to one of the four instances. For each instance, three increasingly difficult target values are used. In each row of the figure, the difficulty of finding a solution with a given target value increases from left to right.

The estimated parameters for all GRASPs are shown in Tables 2– 6. Parameter $\hat{\lambda}$ is the estimated mean time to target solution and parameter $\hat{\mu}$ is the estimated minimum time to target solution. Since the minimum time to target solution is the time corresponding to one GRASP iterations, $\hat{\mu}$ is an estimate of one GRASP iteration.

Figures 7 and 8 show, respectively, the Q-Q plots and superimposed empirical and theoretical distributions for the maximum independent set instances. For the quadratic assignment problem instances, the Q-Q plots and superimposed empirical and theoretical distributions are shown, respectively, in Figures 9 and 10. Figures 11 and 12 depict, respectively, the Q-Q plots and superimposed empirical and theoretical distributions for the graph planarization instances. For the maximum satisfiability instances, the Q-Q plots and superimposed empirical and theoretical distributions are depicted, respectively, in Figures 13 and 14. Figures 15 and 16 show, respectively, the Q-Q plots and superimposed empirical and theoretical distributions for the maximum covering instances.

We make the following observations regarding the experiments.

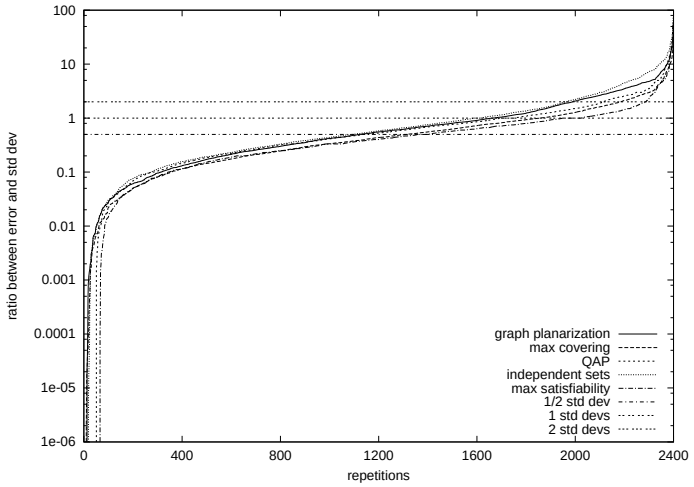

FIGURE 5. Ratio of error and standard deviation for all repetitions – All instances.

12,000 independent runs were carried out, each finding a solution at least as good as its respective target.

In general, there was no large or systematic departure from straightness in the Q-Q plots. However, it is well known in statistics that the samples of real data are often contaminated by a small fraction of abnormal observations that will lie outside the typical range of data. This can be observed in the experiments reported here.

Straightness generally increased with problem difficulty, i.e. as the target value approached the optimal, the fit of solution time to the theoretical distribution improved, implying therefore that the computed parameters were good estimates. This occurs because the distribution of number of GRASP iterations until target solution is more spread out. In some of the easier instances, many runs took few iterations. We further discuss this issue later in this section.

The points in each Q-Q plot can be regarded as order statistics. Due to the high correlation that exists between neighboring order statistics, the probability that a particular point deviates from the line by more than two standard deviations is small. However, as commented in Section 3, the probability that at least one of the points deviates from the line by two standard deviations is undoubtedly much greater. Figures 5 and 6 plot the ratios of deviation from the fitted line to one standard deviation, for all 2400 instances of each GRASP and all 800 harder instances of each GRASP, respectively. The harder instances correspond to those in the rightmost column of the Q-Q plots and superimposed plots of the empirical and theoretical distributions. The ratios are sorted in increasing order. About 75% of all points in the Q-Q plots fall within one standard deviation of the fitted line and about 88% fall within two standard deviations. When limited only to hard instance/target pairs, then about 80% of the all points in the Q-Q plots fall within one standard deviation of the fitted line and about 93% fall within two standard deviations.

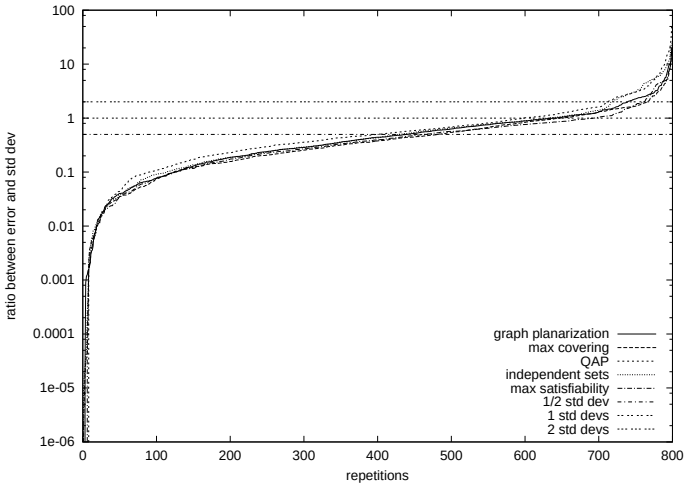

FIGURE 6. Ratio of error and standard deviation for repetitions – Only harder instances.

As can be seen in the Q-Q plots, not many points associated with large CPU times fall outside the one standard deviation bounds. Hence, most of the points that fall outside the two standard deviation bounds are points associated with small CPU times which in turn have small standard deviations. In fact, if we only consider the largest 180 (of 200) CPU times for each instance/target pair, we observe that 93% of all points in the Q-Q plots fall within two standard deviations of the fitted line. Restricting our sample to only hard instance/target plots, then 98% of the points fall within the bounds.

A quantile-quantile plot with horizontal segments, as observed in Figures 7 and 11, is common in practice [4]. This discrete granularity may mean that the data were rounded at some earlier stage before being plotted. In our experiments, this is observed only for the easiest problem/target pairs. It occurs because in many runs, the target solution was found in the same GRASP iteration. In these examples, the i-th horizontal segment depicts the solution times found in the i-th GRASP iteration. Although this is a departure from normality, if these repetitions were eliminated and each segment turned into a single point, represented by its median, nearly straight lines would be observed for all these plots.

5. Concluding remarks

Though it is clear that distributing the GRASP iterations evenly among parallel processors achieves linear speedup for total time (to run all GRASP iterations), it is less clear why linear speedup in time to a given target value is frequently observed. If time to a target solution value fits a two-parameter exponential distribution, then the probability of finding a solution of a given value in time ρt with a sequential process is equal to the probability of finding a solution at least as good as that given value in time t with ρ independent parallel processes. Hence, linear speedup would be observed.

In this paper, we reported on an empirical investigation of the distribution of solution time to a target value. 12,000 GRASP runs were done in the experiment.

To analyze the observations, we used a standard graphical methodology for data analysis. Q-Q plots with variability information were used to determine if the empirical distribution fits its theoretical counterpart. The estimated parameters of the theoretical distribution were derived from the Q-Q plots.

The main conclusion from the experiments is that time to target value indeed fits well a two-parameter exponential distribution. The fit tends to improve as the difficulty to find a solution of a given target value increases.

Though this study was limited to five distinct GRASPs, we believe that this characteristic is present in any GRASP implemented in a straightforward manner. It should be noted that tricks commonly used to speedup a sequential GRASP, such as hash tables to avoid repetition of local search from identical starting solutions, can make speedup in time to target solution be sublinear in the number of processors. An example of this can be seen in Martins, Resende, Ribeiro, and Pardalos [17], where the use of a hash table improves the speed of a sequential GRASP in instances in which the construction phase generates many identical solutions and the parallel GRASP repeats many of these unnecessary local searches.

Acknowledgment

R.M. Aiex was supported by the Brazilian Council for Scientific and Technological Development (CNPq) and was done while this author was visiting AT&T Labs Research.

References

[1] R. Battiti and G. Tecchiolli. Parallel biased search for combinatorial optimization: Genetic algorithms and TABU. *Microprocessors and Microsystems*, 16:351–367, 1992.

[2] B. Bollobás. *Random graphs*. Academic Press, 1985.

[3] R. Burkard, S. Karisch, and F. Rendl. QAPLIB – A quadratic assignment problem library. *European Journal of Operations Research*, 55:115–119, 1991.

[4] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical Methods for Data Analysis*. Chapman & Hall, 1983.

[5] R.J. Cimikowski. An analysis of heuristics for the maximum planar subgraph problem. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 322–331, 1995.

[6] N. Dodd. Slow annealing versus multiple fast annealing runs: An empirical investigation. *Parallel Computing*, 16:269–272, 1990.

[7] H.M.M. Ten Eikelder, M.G.A. Verhoeven, T.W.M. Vossen, and E.H.L. Aarts. A probabilistic analysis of local search. In I.H. Osman and J.P. Kelly, editors, *Metaheuristics: Theory & applications*, pages 605–618. Kluwer Academic Publishers, 1996.

[8] T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

[9] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[10] T.A. Feo, M.G.C. Resende, and S.H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878, 1994.

[11] P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. Technical report, AT&T Labs Research, Florham Park, NJ 07733, 2000.

[12] C. Fleurent and F. Glover. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, 11:198–204, 1999.

[13] O. Goldschmidt and A. Takvorian. An efficient graph planarization two-phase heuristic. *Networks*, 24:69–73, 1994.

[14] J.P. Hart and A.W. Shogan. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6:107–114, 1987.

[15] H.H. Hoos and T. Stützle. Towards a characterisation of the behaviour of stochastic local search algorithms for sat. *Artificial Intelligence*, 112:213–232, 1999.

[16] Y. Li, P.M. Pardalos, and M.G.C. Resende. A greedy randomized adaptive search procedure for the quadratic assignment problem. In P.M. Pardalos and H. Wolkowicz, editors, *Quadratic assignment and related problems*, volume 16 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 237–261. American Mathematical Society, 1994.

[17] S.L. Martins, M.G.C. Resende, C.C. Ribeiro, and P.M. Pardalos. A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. *Journal of Global Optimization*, 2000. To appear.

[18] S.L. Martins, C.C. Ribeiro, and M.C. Souza. A parallel GRASP for the Steiner problem in graphs. In A. Ferreira and J. Rolim, editors, *Proceedings of IRREGULAR'98 – 5th International Symposium on Solving Irregularly Structured Problems in Parallel*, volume 1457 of *Lecture Notes in Computer Science*, pages 285–297. Springer-Verlag, 1998.

[19] R.A. Murphey, P.M. Pardalos, and L.S. Pitsoulis. A parallel GRASP for the data association multidimensional assignment problem. In P.M. Pardalos, editor, *Parallel processing of discrete problems*, volume 106 of *The IMA Volumes in Mathematics and Its Applications*, pages 159–180. Springer-Verlag, 1998.

[20] L.J. Osborne and B.E. Gillett. A comparison of two simulated annealing algorithms applied to the directed Steiner problem on networks. *ORSA J. on Computing*, 3:213–225, 1991.

[21] P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A parallel GRASP implementation for the quadratic assignment problem. In A. Ferreira and J. Rolim, editors, *Parallel Algorithms for Irregularly Structured Problems – Irregular'94*, pages 111–130. Kluwer Academic Publishers, 1995.

[22] P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A parallel GRASP for MAX-SAT problems. *Lecture Notes in Computer Science*, 1184:575–585, 1996.

[23] P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. Algorithm 769: Fortran subroutines for approximate solution of sparse quadratic assignment problems using GRASP. *ACM Transactions on Mathematical Software*, 23:196–208, 1997.

[24] M.C. Rangel, N.M.M. Abreu, and P.O. Boaventura Netto. GRASP in the QAP: An acceptance bound for initial solution. In *Proc. of the Third Metaheuristics International Conference*, pages 381–386, July 1999.

[25] M.C. Rangel, N.M.M. de Abreu, P.O. Boaventura Netto, and M.C.S. Boeres. A modified local search for GRASP in the quadratic assignment problem. Technical report, Production Engineering Program, COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, RJ Brazil, 1998.

[26] M.G.C. Resende. Computing approximate solutions of the maximum covering problem using GRASP. *J. of Heuristics*, 4:161–171, 1998.

[27] M.G.C. Resende and T.A. Feo. A GRASP for satisfiability. In D.S. Johnson and M.A. Trick, editors, *Cliques, Coloring, and Satisfiability: The Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 499–520. American Mathematical Society, 1996.

[28] M.G.C. Resende, T.A. Feo, and S.H. Smith. Algorithm 787: Fortran subroutines for approximate solution of maximum independent set problems using GRASP. *ACM Trans. Math. Software*, 24:386–394, 1998.

[29] M.G.C. Resende, P.M. Pardalos, and Y. Li. Algorithm 754: Fortran subroutines for approximate solution of dense quadratic assignment problems using GRASP. *ACM Transactions on Mathematical Software*, 22:104–118, 1996.

[30] M.G.C. Resende, L.S. Pitsoulis, and P.M. Pardalos. Approximate solution of weighted MAX-SAT problems using GRASP. In J. Gu and P.M. Pardalos, editors, *Satisfiability problems*, volume 35 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 393–405. American Mathematical Society, 1997.

[31] M.G.C. Resende, L.S. Pitsoulis, and P.M. Pardalos. Fortran subroutines for computing approximate solutions of MAX-SAT problems using GRASP. *Discrete Applied Mathematics*, 100:95–113, 2000.

[32] M.G.C. Resende and C.C. Ribeiro. A GRASP for graph planarization. *Networks*, 29:173–189, 1997.

[33] C.C. Ribeiro and M.G.C. Resende. Algorithm 797: Fortran subroutines for approximate solution of graph planarization problems using GRASP. *ACM Transactions on Mathematical Software*, 25:341–352, 1999.
[34] B. Selman, H.A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the AAAI-94*, pages 337–343. MIT Press, 1994.
[35] E.D. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.
[36] M.G.A. Verhoeven and E.H.L. Aarts. Parallel local search. *J. of Heuristics*, 1:43–66, 1995.

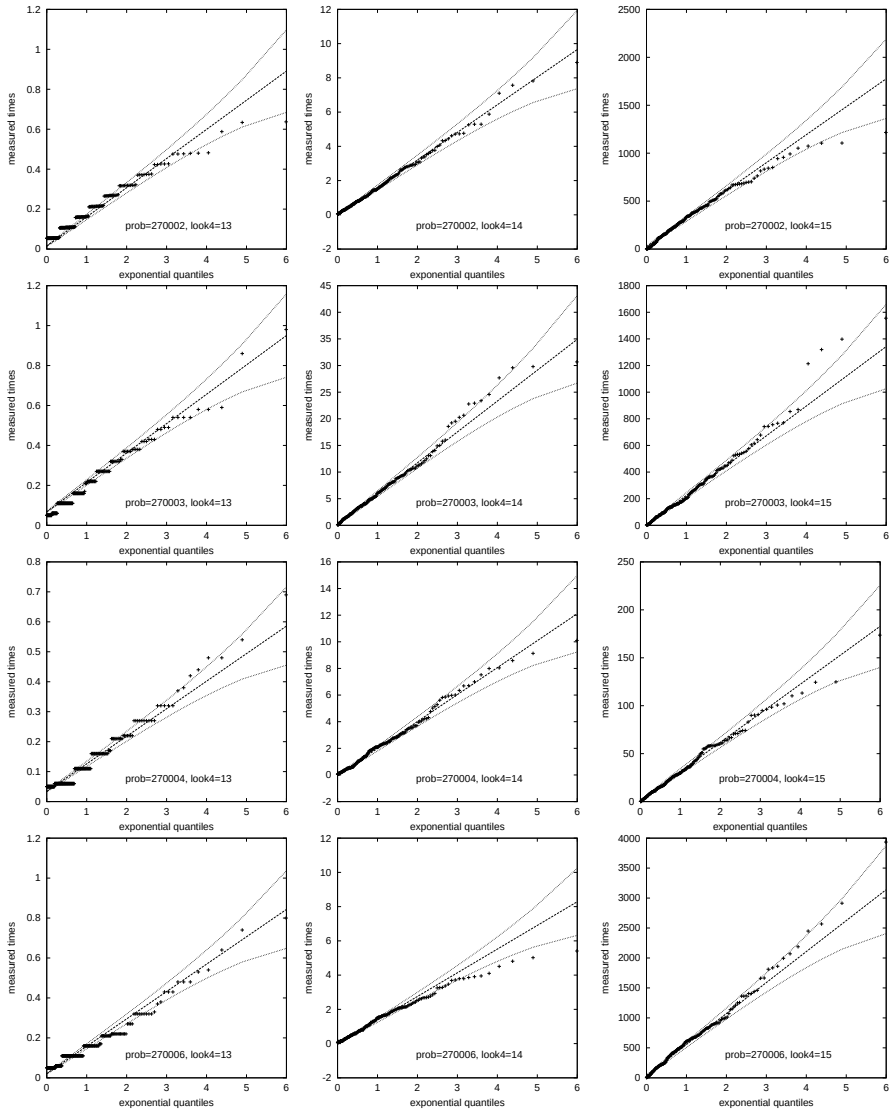

FIGURE 7. Q-Q plots for GRASP for maximum independent set

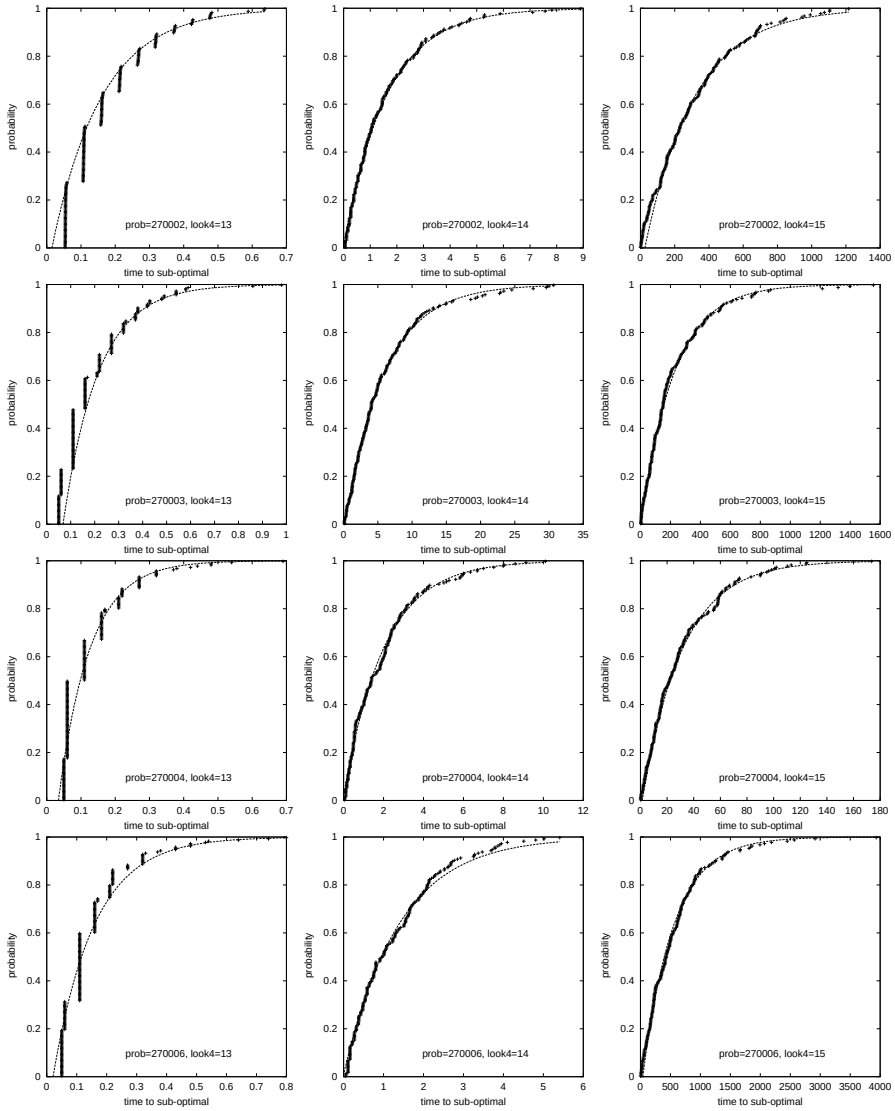

FIGURE 8. Exponential plots for GRASP for maximum independent set

(R.M. Aiex) DEPARTMENT OF COMPUTER SCIENCE, CATHOLIC UNIVERSITY OF RIO DE JANEIRO, R. MARQUÊS DE SÃO VICENTE, 225, RIO DE JANEIRO, RJ 22453-900 BRAZIL
E-mail address, R.M. Aiex: rma@inf.puc-rio.br

(M.G.C. Resende) INFORMATION SCIENCES RESEARCH, AT&T LABS RESEARCH, 180 PARK AVENUE, ROOM C241, FLORHAM PARK, NJ 07932 USA.
E-mail address, M.G.C. Resende: mgcr@research.att.com

(C.C. Ribeiro) DEPARTMENT OF COMPUTER SCIENCE, CATHOLIC UNIVERSITY OF RIO DE JANEIRO, R. MARQUÊS DE SÃO VICENTE, 225, RIO DE JANEIRO, RJ 22453-900 BRAZIL
E-mail address, C.C. Ribeiro: celso@inf.puc-rio.br

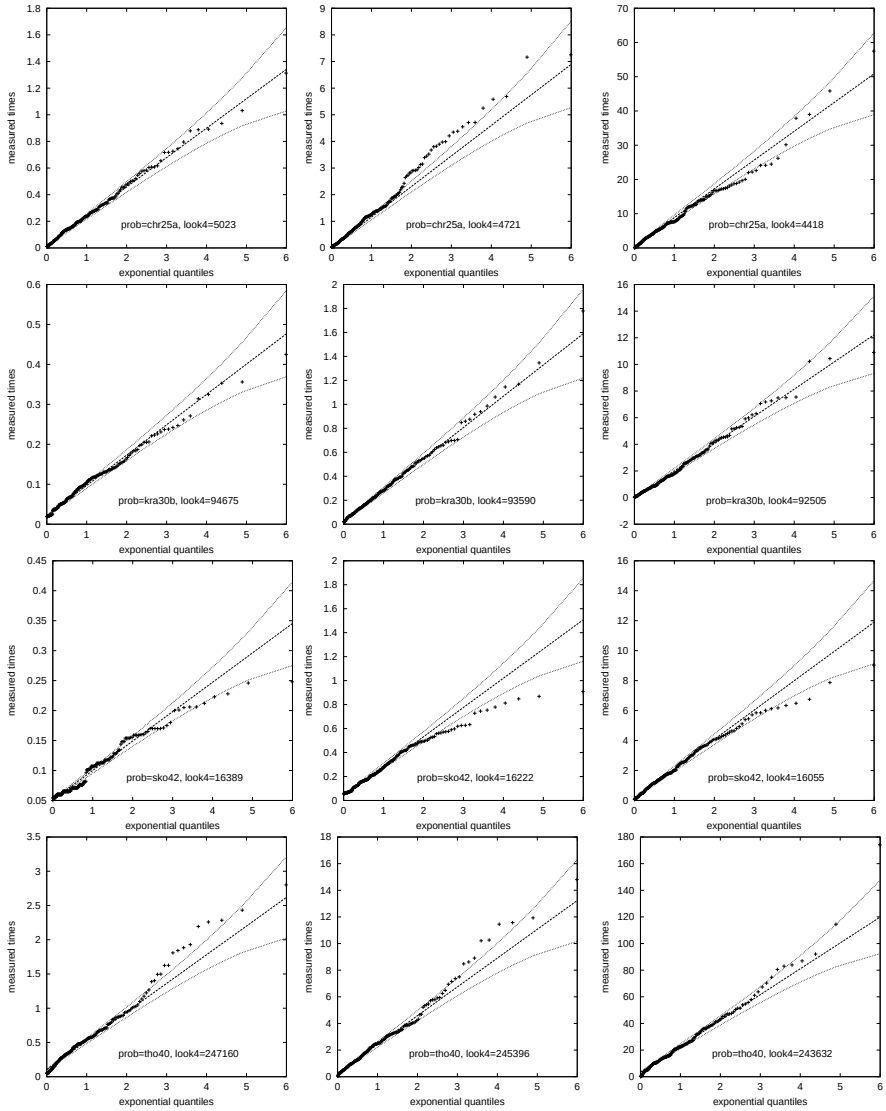

FIGURE 9. Q-Q plots for GRASP for quadratic assignment

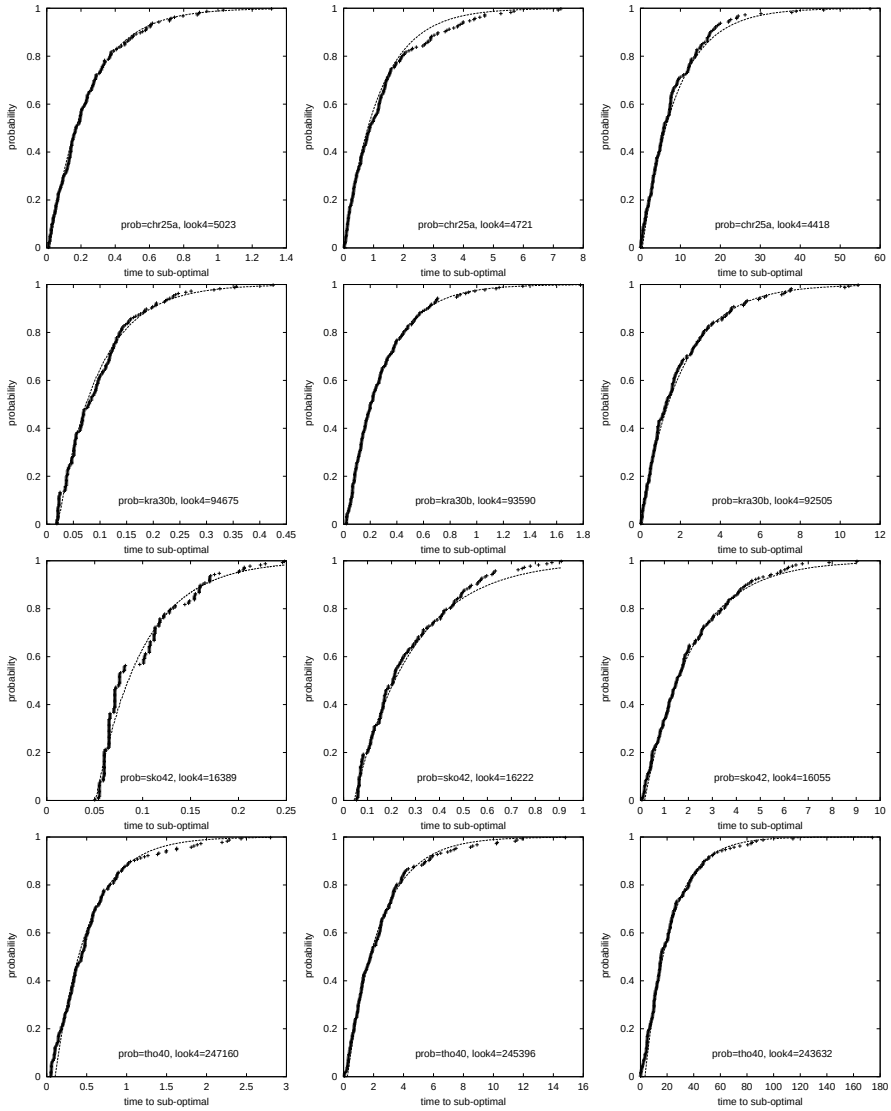

FIGURE 10. Exponential plots for GRASP for quadratic assignment

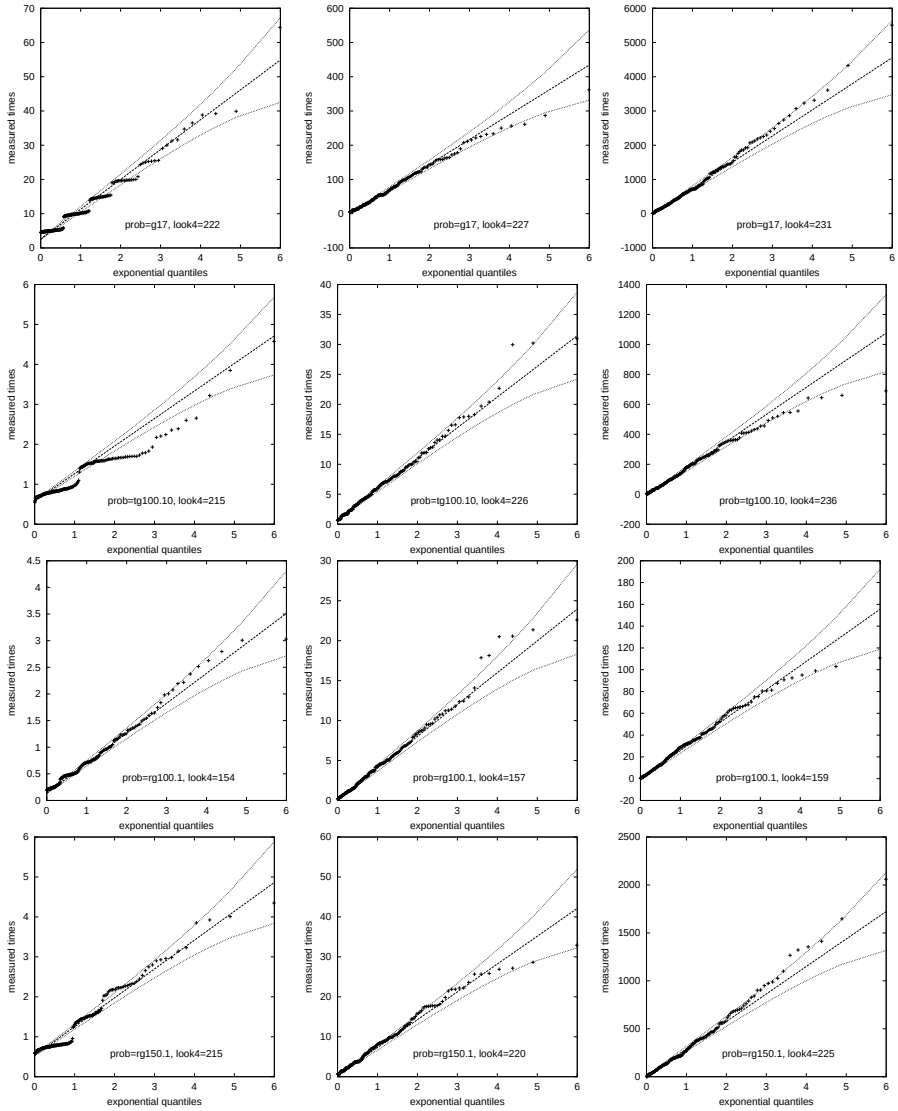

FIGURE 11. Q-Q plots for GRASP for graph planarization

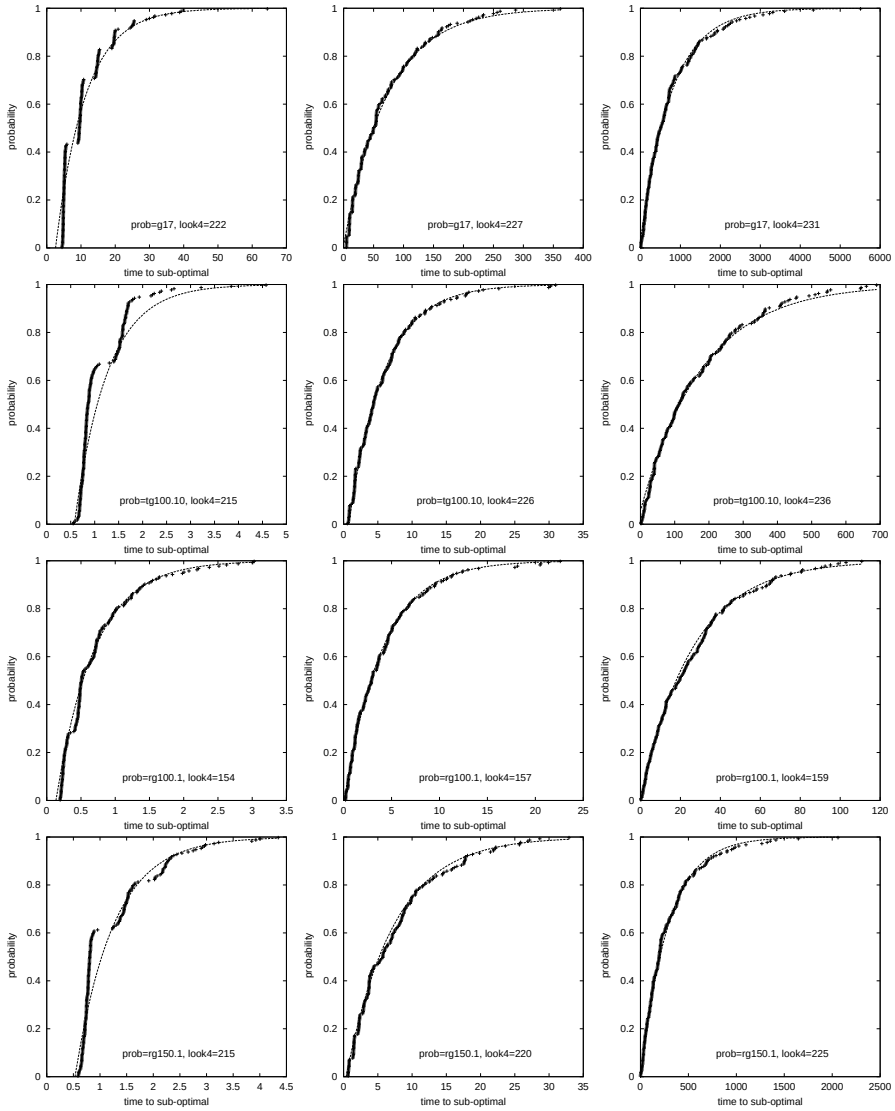

FIGURE 12. Exponential plots for GRASP for graph planarization

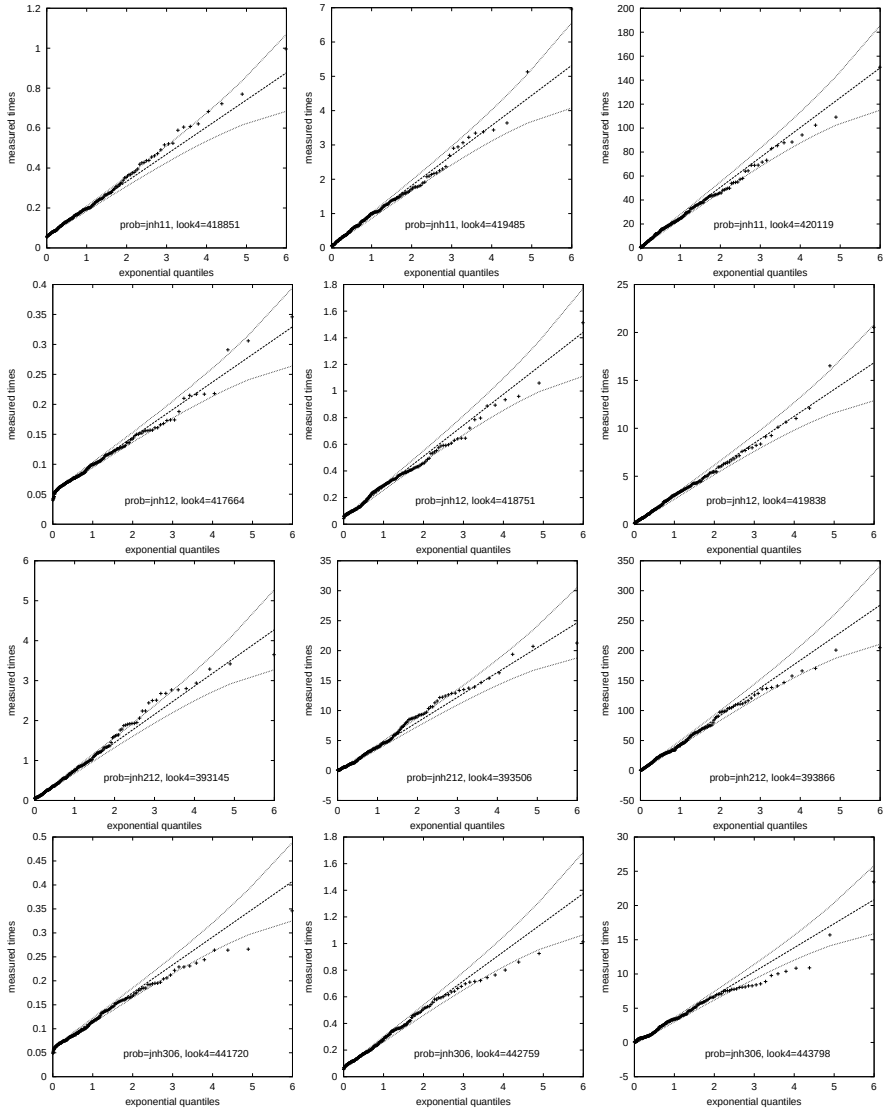

FIGURE 13. Q-Q plots for GRASP for maximum weighted satisfiability

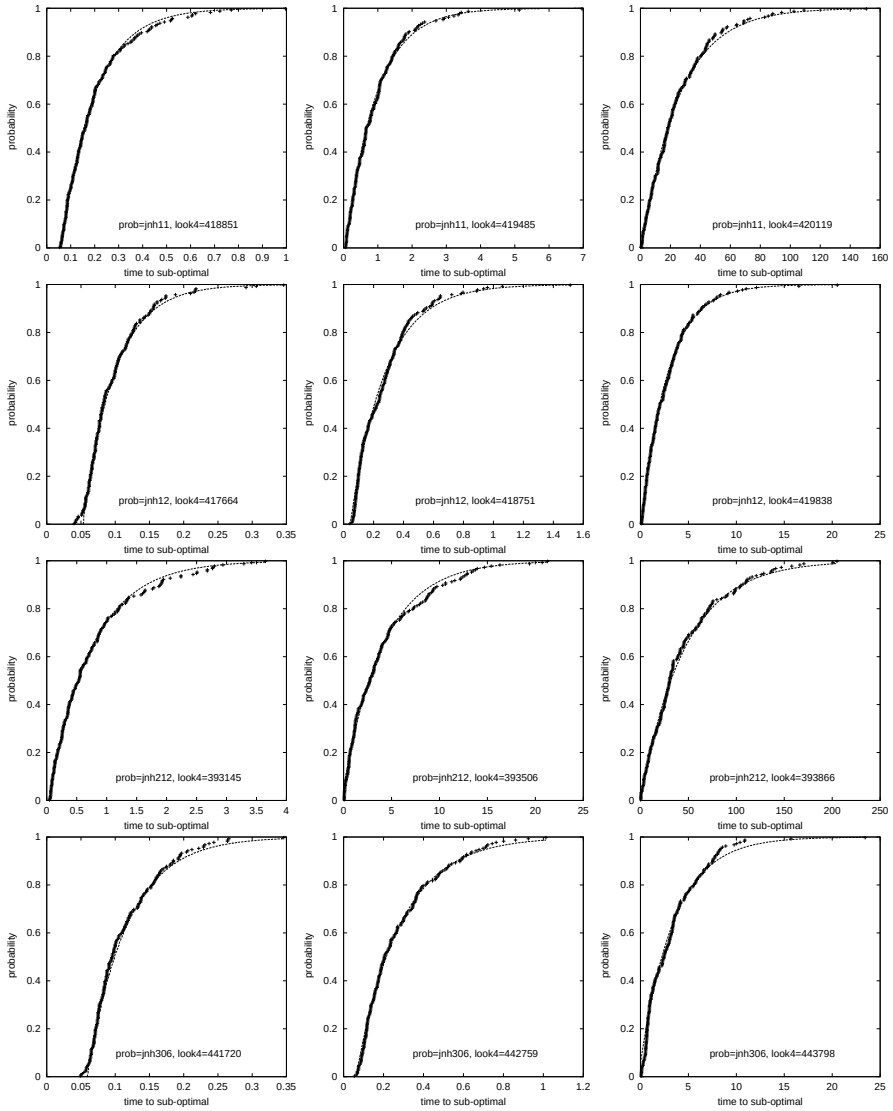

FIGURE 14. Exponential plots for GRASP for maximum weighted satisfiability

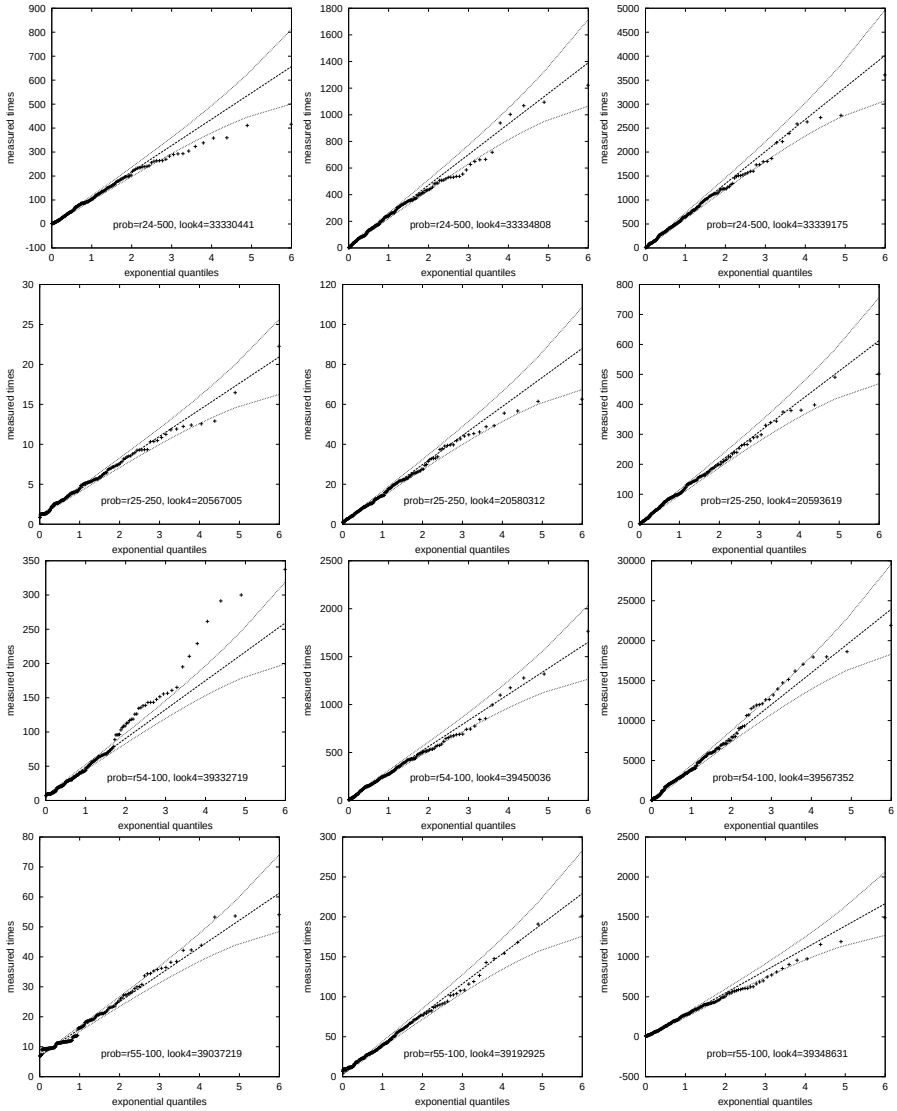

FIGURE 15. Q-Q plots for GRASP for maximum covering

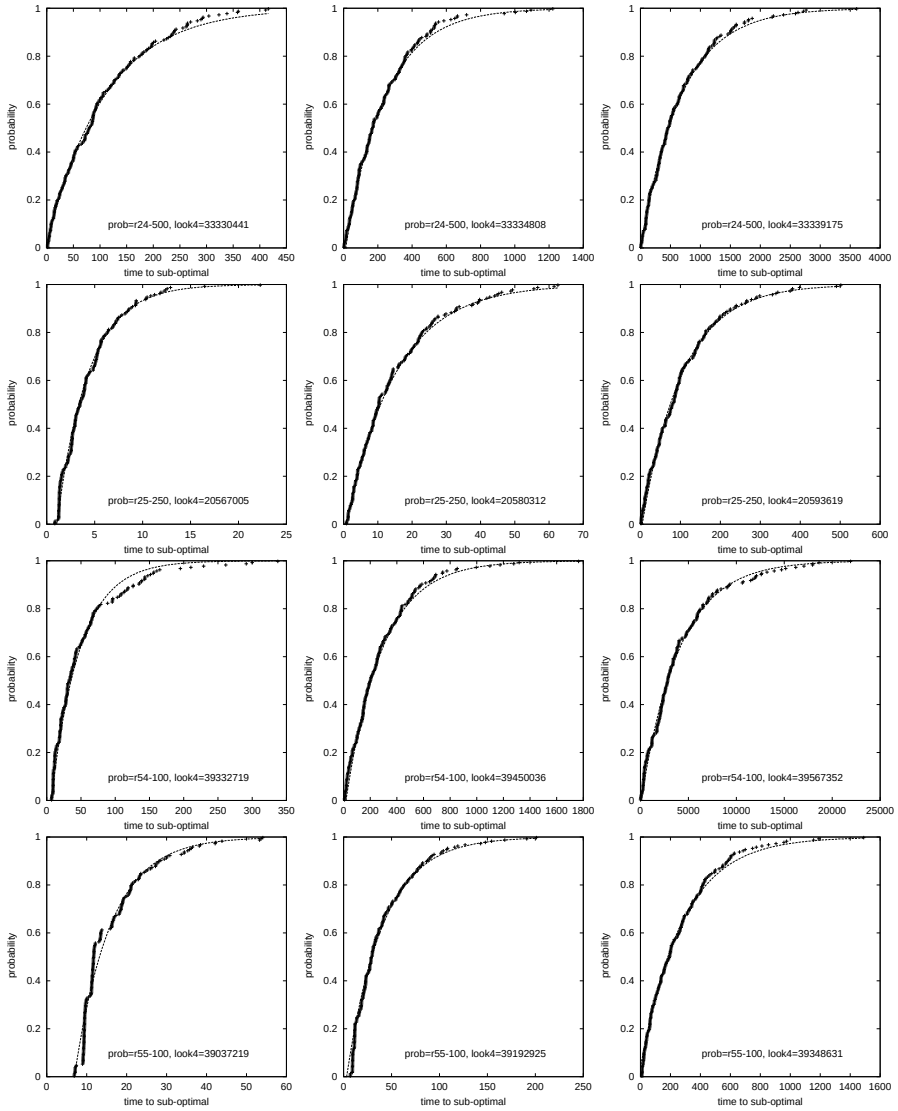

FIGURE 16. Exponential plots for GRASP for maximum covering