Biased Randomization of Heuristics using Skewed Probability Distributions: applications to routing and other problems

Grasas A, Juan AA, Faulin J, de Armas J, Ramalhinho H. Biased randomization of heuristics using skewed probability distributions: a survey and some applications. Comput Ind Eng. 2017 Jun 10;110:216-28.

Abstract:

Randomized heuristics are widely used to solve large scale combinatorial optimization problems. Among the plethora of randomized heuristics, this paper reviews those that contain biased-randomized procedures (BRPs). A BRP is a procedure to select the next constructive 'movement' from a list of candidates in which their elements have *different* probabilities based on some criteria (e.g., ranking, priority rule, heuristic value, etc.). The main idea behind biased randomization is the introduction of a slight modification in the greedy constructive behavior that provides a certain degree of randomness while maintaining the logic behind the heuristic. BRPs can be categorized into two main groups according to how choice probabilities are computed: *(i)* BRPs using an empirical bias function; and *(ii)* BRPs using a skewed theoretical probability distribution. This paper analyzes the second group and illustrates, throughout a series of numerical experiments, how these BRPs can benefit from parallel computing in order to significantly outperform heuristics and even simple metaheuristic approaches, thus providing reasonably good solutions in 'real time' to different problems in the areas of transportation, logistics, and scheduling.

Keywords: Heuristics, Biased Randomization, Real-time Decision Making, Combinatorial Optimization, Logistics, Transportation, Production.

1. Introduction

A number of complex decision-making processes in real-life transportation, logistics, and production systems can be modeled as combinatorial optimization problems (Faulin *et al*, 2012). Among many others, some typical examples include: vehicle routing problems (VRP) (Toth and Vigo, 2014), arc routing problems (Corberán and Laporte, 2014), facility location problems (Chan, 2011), or scheduling problems (Pinedo, 2012). All these problems are NP-hard in nature, meaning that the space of potential solutions grows very fast (exponential explosion) as the instance size increases. Therefore, using exact methods is not always the most efficient strategy, especially when the size of the problem instance is large and reasonably good decisions are needed in negligible computing times. Under these circumstances, heuristic-based approaches constitute an excellent alternative to exact methods (Talbi, 2009). Accordingly, a large number of heuristic and metaheuristic algorithms have been developed during the last decades to solve large scale combinatorial optimization problems and, eventually, support intelligent decision-making processes in a myriad of fields, including transportation, logistics, production, finance, telecommunication, Internet computing, health care, etc.

A constructive heuristic is a computational method that employs an iterative process to generate a feasible solution of reasonable quality. At each iteration of the solution-building process, the next 'movement' is selected from a list of potential candidates that has been sorted according to some criteria. *Pure greedy heuristics* always select the next 'most promising' movement. As a result, these heuristics are expected to generate a reasonably good solution once the entire list is traversed. Notice, however, that this is a somewhat myopic behavior, since the heuristic selects the next movement without considering how the current selection will affect subsequent decisions as the list is processed downwards. Even worse, this property results in a deterministic procedure, i.e., the same solution is obtained every time the algorithm is run. Examples of such methods are the nearest neighbor for traveling salesman problems (Lawler *et al*, 1985), the shortest processing time dispatching rule for scheduling problems (Pinedo and Chao, 1999), or the savings algorithm for VRPs (Clarke and Wright, 1964). Although these methods are easy to implement and can be run almost instantaneously, the real-time solutions they provide are usually far from being optimal. To improve the quality of these heuristic solutions –and as far as more time is available–, different types of *local search methods* can be used to explore the solution neighborhood (Aarts and Lenstra, 1997). Typically, the neighbor selection is based on a certain logic that tries to take advantage of the specific characteristics of the optimization problem being considered. This usually leads to local optimal solutions. As in the construction phase, if the neighbor chosen is always the next 'most promising' movement according to some criteria, the resulting searching process will be deterministic too.

Randomization techniques are frequently used to escape from this local optimality trap and improve the overall quality of the solution. These techniques can be incorporated either in the construction phase and/or the local search. Randomization allows exploring alternative solutions by selecting an element other than the 'most promising option on the short run'. This leads to different outputs each time the entire procedure is executed. Since running a heuristic might take only a few seconds –or even less in a modern computer if the heuristic is correctly implemented and the instance size is not extremely large–, one can execute it several times, either in sequential mode or in parallel mode by using different threads, and then select the best of the stochastic outputs. Countless metaheuristic algorithms include uniform randomization in their procedures. However, a uniform randomization of the list of candidate elements destroys the logic behind the heuristic greedy behavior. In order to maintain this logic, the randomization can be biased (i.e., oriented) so that higher probabilities are given to the most promising candidates. Thus, the main idea behind biased randomization is the introduction of a slight modification in the greedy constructive behavior that provides a certain degree of randomness while maintaining the main logic behind the heuristic. In a seminal paper on the Monte Carlo method, King (1953) already emphasized the enormous improvement of biasing probabilities on sampling efficiency. Different methods to bias the randomization have been used in multiple contexts thereafter (Figure 1). Among them, this paper pays special attention to the ones that use skewed (non-symmetric) theoretical probability distributions in order to introduce an appropriate bias in the process of selecting elements from the list during the constructive and/or local search stages. Some skewed theoretical distributions, such as the geometric or the decreasing triangular ones, offer at least two advantages over using empirical distributions: *(i)* they contain at most one simple parameter, which can be easily set; and *(ii)* they can be sampled using well-known analytical expressions, which from a computational perspective is typically faster than other sampling techniques involving the use of loops.

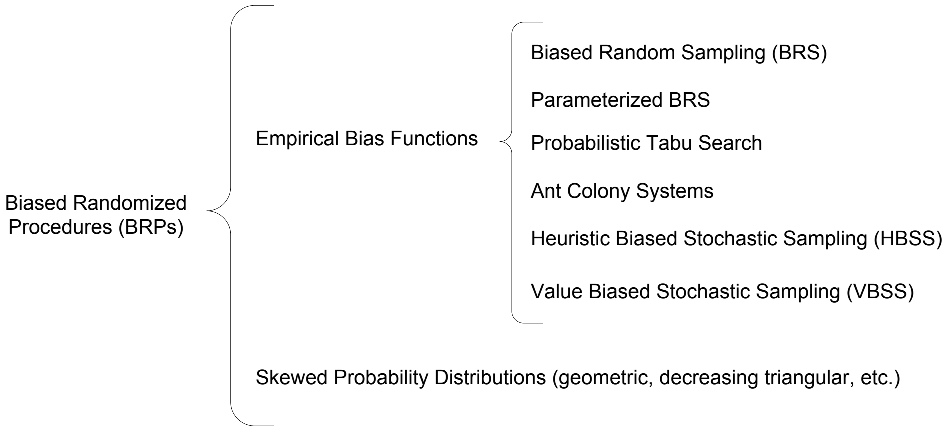
Biased Randomized Procedures (BRPs)
Empirical Bias Functions
Biased Random Sampling (BRS)
Parameterized BRS
Probabilistic Tabu Search
Ant Colony Systems
Heuristic Biased Stochastic Sampling (HBSS)
Value Biased Stochastic Sampling (VBSS)
Skewed Probability Distributions (geometric, decreasing triangular, etc.)

Figure 1: A classification of Biased Randomized Procedures.

In particular, the main contributions of this paper are: *(i)* to provide a review of the most relevant *biased randomized procedures* (BRPs) used in the literature to solve combinatorial optimization problems; *(ii)* to provide a general framework for BRPs that use a skewed theoretical probability distribution to bias the selection of the next movement during the constructive and/or local search processes; and *(iii)* to illustrate, throughout a series of numerical experiments, how these BRPs can significantly outperform heuristics, and even simple metaheuristic approaches, thus providing reasonably good solutions in 'real time' (e.g., one or two seconds) to different transportation, logistics, and scheduling problems.

The remainder of this paper is structured as follows: Section 2 introduces the concept of randomized algorithms; Section 3 presents different BRPs that use empirical bias functions; Section 4 provides a general framework for BRPs with a skewed theoretical probability distribution, and discusses the advantages of this approach over the one based on empirical bias functions; Section 5 analyzes different applications of BRPs to the fields of logistics, transportation, and scheduling; Section 6 describes a series of computational experiments that contribute to illustrate and quantify the potential of BRPs; finally, Section 7 summarizes the main contributions of the paper.

2. Randomized Algorithms

There is an enormous body of literature that study probabilistic or randomized algorithms and a review of that is far beyond the scope of this paper. The reader is referred to Collet and Rennard (2006) for a review, and to Clerc (2015) for a vast discussion about the stochastic aspects of optimization. The focus of this paper is in the subset of randomized algorithms that include some type of bias in any of their random processes. A randomized algorithm uses random bits to make random choices during its execution. Unlike deterministic algorithms, different solutions are obtained every time the procedure is executed. The most successful approaches to solve large combinatorial problems take advantage of this feature to perform several iterations and collect the best overall output. These approaches are commonly known as *multi-start methods* (Martí *et al*, 2013). In general terms, they all contain two differentiated phases: a construction process and a local or neighborhood search. The former diversifies the search for solutions while the latter intensifies this search. These two phases are repeated until a stopping criterion is satisfied. Note that the randomized procedure can be applied at either phase because there is always a discrete choice that has to be made.

Many randomized procedures found in the literature rely on uniform randomization, that is, they use the uniform probability distribution when selecting an element, neighbor, or solution. These could be categorized as *uniformly-randomized algorithms*. The main drawback of such approaches is that they do not benefit from the heuristic 'common sense': if candidate elements are ranked according to their 'goodness' on a given criterion, choosing one via a uniform random process fades away the advantages of the sorting. This is partially overcome by *partially-randomized algorithms*, that use uniform randomization but on a subset of candidates. The greedy randomized adaptive search procedure (GRASP) is the most representative and commonly used algorithm of this type. It was initially proposed by Feo and Resende (1995) and extensively used in multiple applications (Resende and Ribeiro, 2010). As a multi-start method, each GRASP iteration is composed of a construction phase and a local search. In the construction phase, all candidate elements are sorted according to a greedy evaluation function. The 'best next' elements, i.e., those whose addition represents the highest improvement on the objective function, are added to a restricted candidate list (RCL). The next element is chosen randomly (using a uniform distribution) from this RCL and added to the partial solution. This is performed iteratively until there are no more candidates. A local search is then applied until a local optimal is reached. Since only the elements in the RCL can be included in the partial solution, this method can be seen as a multi-start method with a partially random construction heuristic. A similar idea is the *window random sampling* by Valls *et al* (2003), used in a resource-constrained project scheduling problem. A *window* parameter is defined as the maximum difference allowed between the order of the candidate and the minimal order. In the field of genetic algorithms, there also exists heuristics with randomization partially guided according to some criteria. This is the case, for instance, of the biased random key genetic algorithm (Gonçalves and Resende, 2011). In this genetic algorithm, the population is partitioned into two groups: elite and non-elite individuals. When this population is evolved to obtain the next generation, some of the children are obtained by the process of mating two randomly selected individuals, one from the elite group and one from the non-elite group. Mating is done using parameterized uniform crossover, that is, the genes from the elite parents have larger probability of being selected. This way, the randomization is partially biased to favor elite parent's characteristics over the non-elite parent's ones.

3. BRPs using Empirical Bias Functions

A biased (oriented) randomized procedure aims at selecting the next element while capturing the best of two realms: exploitation and exploration. On the one hand, the procedure favors the most promising candidates to exploit the solution space; on the other hand, it introduces a weighted randomness degree to explore this solution space. To determine the balance between either one, a BRP may use a *bias function*. A bias function, $bias(\cdot)$, is a function that assigns a non-negative weight to all elements in the candidate list. These weights are then normalized to obtain an empirical probability distribution that will define the set of probabilities. Note that two extreme cases can be constructed by giving the same weight to all elements (uniform distribution), or by giving a positive weight to the top element and zero weight to the rest (greedy). All other weight allocations provide intermediate bias configurations. This section presents different BRPs that include some sort of bias function to build the empirical probability distribution. The bias function is therefore an algorithm input that must be designed considering both the problem characteristics as well as the responsiveness of the chosen heuristic –in terms of performance– to different types of bias.

3.1 Biased Random Sampling

Biased random sampling (BRS) was one of the earliest BRPs employed in the literature. In these early heuristics, some specified criteria were used to bias the choice of randomly generated solutions. To the best of our knowledge, the first heuristics that incorporated BRS were used to solve assembly line balance problems (Arcus, 1965; Tonge, 1965), production scheduling problems (Giffler *et al*, 1963; Heller and Logeman, 1962), location problems (Mabert and Whybark, 1977; Nugent *et al*, 1968), and an inventory management problem (Berry *et al*, 1977).

3.2 Parameterized Biased Random Sampling

Parameterized BRS is a randomized method in which the probability values to select the next candidates are biased according to priority rules. Numerous priority rule-based heuristics have been designed to tackle resource-constrained project scheduling problems. For this vastly studied problem, Cooper (1976) presented the first BRS scheme that used nine different priority rules as weighting factor to bias the probability of choosing an activity. This probability was calculated by dividing the activity priority value by the sum of the priority values of all activities in the candidate list. Later, Drexl (1991) introduced *regret based biased random sampling*. This sampling technique uses the priority values indirectly via *regret* values. The regret of a job is the difference between its priority value and the lowest overall priority value. Probabilities are then calculated using a parameter that controls the bias degree. Schirmer and Riesenberg (1997) proposed BRS variants, dubbed the *normalized BRS* and the *modified regret based BRS*, to cope with some of the drawbacks of the existing sampling approaches. The authors stated that, in general, they always outperformed uniform random sampling approaches. In a similar line of research, Valls *et al* (2003) developed the *β-BRS* method. The β parameter establishes the probability of choosing the activity on top of the priority-rule based list. Lastly, Coelho and Tavares (2003) designed the *global BRS* method. Unlike previous sampling schemes, this one perturbs the priority values by adding a random value between 0 and 1. Activities are then scheduled in the order defined by the modified priority list. The reader is referred to Kolisch and Hartmann (1999) for a summary of some of these sampling techniques in the resource-constrained project scheduling problem.

3.3 Probabilistic Tabu Search

For search heuristics, Glover (1989) introduced the first big family of metaheuristics, the probabilistic tabu search (PTS), which incorporated a BRP –usually in the move acceptance function. Tabu search (Glover, 1990) is a “higher level heuristic designed to guide other methods to escape the trap of local optimality”. PTS is an extension of tabu search that includes a skewed probabilistic element within the search. Biasing is a way to control the diversity in the search, and can be achieved by considering: *(i)* move attractiveness (i.e., the change in the objective function); *(ii)* tabu status (i.e., tenure on a tabu list); and/or *(iii)* aspiration level (i.e., the objective function value in relation to a historical standard). The probabilistic nature of the approach can be a substitute for memory when it is unclear how memory can be used to enhance the result. Some years later, Løkketangen and Glover (1996) adapted PTS to zero-one mixed integer programming problems with probabilistic measures that were both effective and easy to implement.

3.4 Ant Systems and Ant Colony Systems

Another family of probabilistic algorithms that uses a BRP is the ant system (Dorigo *et al*, 1996) and its subsequent variant, the ant colony system (Dorigo and Gambardella, 1997). Inspired by the behavior of real ants, these algorithms mimic the pheromone trails that insects use to establish the shortest paths. The ant system was first applied to the traveling salesman problem. To complete a tour, the cities visited are chosen probabilistically via Monte Carlo sampling. The probabilities in the state transition are biased using the so-called *random-proportional rule* to favor shorter edges with a greater amount of pheromone. The ant colony system includes three main variants, one of which is in the state transition. The modified transition follows the *pseudo-random-proportional rule*. This rule adds a previous step: it randomly selects a number uniformly distributed between 0 and 1; if it is below a given threshold the best edge is selected, otherwise an edge is selected according to the random-proportional rule. By calibrating the threshold, the algorithm determines the relative importance of exploitation (best next edge) versus exploration (biased random edge).

3.5 Reactive GRASP

In the previous section, GRASP was introduced as one of the most well-known partially-randomized algorithms. A variant of GRASP that includes a BRP is the so-called reactive GRASP, first proposed by Prais and Ribeiro (2000). Unlike the original GRASP, the size of the restricted candidate list in a reactive GRASP is not fixed but self-adjusted according to the quality of the solutions found during the search. A BRP is used when selecting the restrictiveness, or size, of the candidate list (i.e., the parameter α). The algorithm starts with a discrete set of predetermined list sizes, α_i. The probability of choosing a given α_i from this set is drawn initially from a uniform distribution. As the algorithm advances, these probabilities are biased using information collected during the search. One possible biasing strategy is to use the average values of the solutions obtained to recompute the probabilities of the different α's. The values that lead to better solutions will be more frequently used in the construction phase of the GRASP procedure.

3.6 Heuristic-Biased Stochastic Sampling

Bresina (1996) devised a BRP called *heuristic-biased stochastic sampling* (HBSS) to solve scheduling problems and other constrained optimization problems. The motivating idea again was to bias the probability of choosing the next partial solution. To avoid a complete random exploration, HBSS uses the search heuristic to guide this exploration. The guidance degree is determined by a given bias function. In the search, the elements of the candidate list are ranked according to the heuristic, and the bias function assigns a weight to each element. These weights are then normalized to be transformed into probabilities. Thus, if r_i denotes the rank of the element e_i, the probability of choosing e_i is given by:

$$P(e_i) = \frac{bias(r_i)}{\sum_j bias(r_j)}$$

Usually, when the heuristic accuracy is high a stronger bias (weight) is set to increase the probabilities of selecting better solutions. On the contrary, when the heuristic accuracy is lower a weaker bias is set to widen the exploration of the solution space. Bresina *et al* (1994) experimented with the following bias functions in the telescope observation scheduling problem:

- Logarithmic: $\text{bias}(r) = \log^{-1}(r+1)$
- Linear: $\text{bias}(r) = 1/r$
- Polynomial (*n* = *2, 3, 4*): $\text{bias}(r) = r^{-n}$
- Exponential: $\text{bias}(r) = e^{-r}$
- Uniform: $\text{bias}(r) = 1$

The best performing bias functions for the particular problem the authors analyzed were the exponential and the second degree polynomial, which were the two functions in the middle in terms of bias strength. The HBSS approach encompasses a family of search algorithms that can be modulated via a bias function ranging from a greedy search to a uniform random search.

3.7 Value-Biased Stochastic Sampling

Following a similar reasoning as in the HBSS, Cicirello and Smith (2005) proposed the *value-biased stochastic sampling* (VBSS) as a search heuristic. In HBSS, the bias function gives weight to the candidates solely based on their rank, ignoring completely their heuristic values. Alternatively, VBSS not only considers rank but it also incorporates the “discriminatory power inherent in the heuristic”. Thus, according to the VBSS approach, the probability of choosing element e_i is given by:

$$P(e_i) = \frac{bias(heuristic(e_i))}{\sum_j bias\left(heuristic(e_j)\right)}$$

The resulting probabilities from both BRPs (VBSS and HBSS), may differ considerably when the choices of the candidate list have very different heuristic values. Cicirello and Smith (2005) provided an illustration of such a case. The authors also tested their approach in the weighted tardiness scheduling problem with sequence-dependent setups. In their experiments, the VBSS approach was able to outperform the HBSS approach.

4. BRPs with Skewed Theoretical Probability Distributions

This section presents a general framework for BRPs that use a skewed theoretical probability distribution to bias the probabilities of the candidate elements. The BRPs considered in the previous section share a common feature: they all rely on some kind of bias function to define the choice probabilities. Using a bias function, each element in the list was assigned a different weight based on some criteria (e.g., ranking, priority rule, heuristic value), and then an empirical probability distribution was built. A random number drawn from this distribution determined the choice of the next element. Alternatively, instead of using an empirically-constructed probability distribution, one could resort directly to a theoretical probability distribution that is already skewed or non-symmetric by definition. Examples of such distributions are the geometric or the decreasing triangular. In particular, in most of our previous work we have used the geometric distribution, since it only has one parameter which determines its specific shape. Also, this parameter varies between 0 and 1, thus facilitating its setting in most practical applications. As values of this parameter get closer to 0, the more uniform-randomized the selection process will be. On the contrary, as these values get closer to 1, the more greedy the selection will be (Juan et *al*, 2010). This type of distributions provides a natural bias for the candidate elements in the list. For instance, Figure 2 compares the probabilities of selecting each of the twenty-five elements of a given sorted list using: a geometric distribution with parameter 0.2 (left part), and a discretized decreasing triangular distribution (right part).

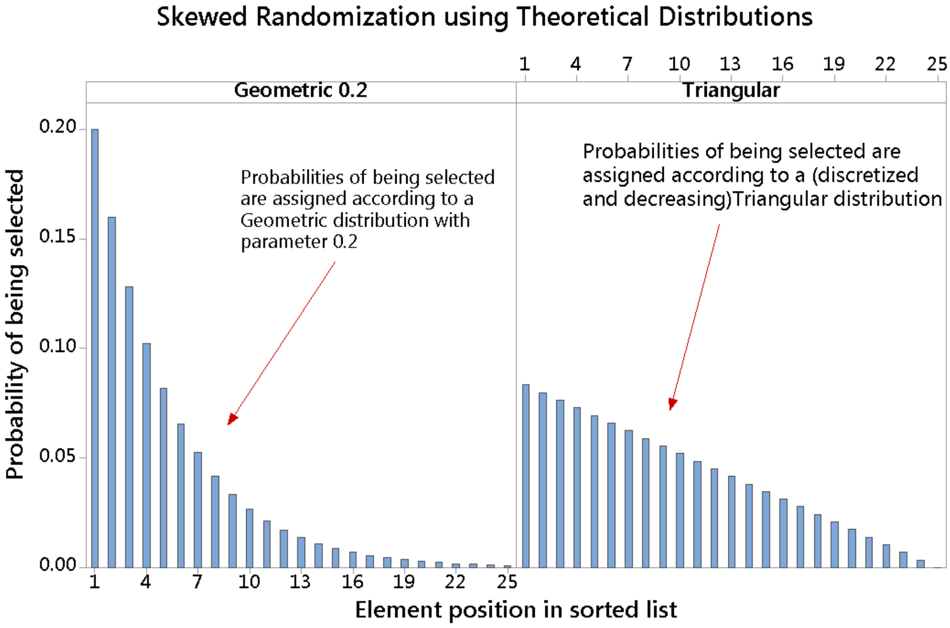

Figure 2: Use of skewed distributions to introduce randomness.

Most heuristics iteratively perform a construction phase followed by a neighborhood search. As discussed above, BRPs can be employed in any of the two stages. Regardless of the stage, there is always a discrete choice that has to be made from a list of potential candidates. Potential candidates could be neighbor solutions, edges in traveling salesman problems and VRPs, jobs or machines in scheduling problems, or tasks or resources in resource-constrained project scheduling problems, for example. The list of candidates is sorted according to a problem-specific criterion, and probabilities are assigned to each element according to a skewed probability distribution. Figure 3 displays a general pseudo-code for a BRP with a skewed distribution. The procedure requires the following inputs: *(i)* the list L of potential candidates, which is sorted according to the criteria provided by the heuristic; *(ii)* the seed `s` for the (uniform) pseudo-random number generator; *(iii)* the skewed probability distribution `PD`; and *(iv)* the parametric values `p` of this distribution. The procedure returns the selected element `l` from the sorted list.

```
Procedure BRP(L, s, PD, p)
    $\mu$ ← using seed s, generate pseudo-random number in [0,1)
    $\rho$ ← using $\mu$, generate random variate from distribution PD(p)
    l ← select the $\rho$-th element of the sorted list L
    return l
End
```

Figure 3: Pseudocode to select the next element using a skewed distribution.

To the best of our knowledge, the first heuristic algorithms including BRPs with a skewed theoretical probability distribution were introduced in Juan *et al* (2010, 2011) in order to solve the VRP. The authors developed a hybrid algorithm that combined the classical savings heuristic (Clarke and Wright, 1964) with Monte Carlo simulation. At each step of the solution-construction process, eligible edges were sorted in a non-increasing savings list. Edges enjoyed (quasi-) exponentially diminishing probabilities that were variable and based upon a geometric distribution. Consequently, the next element was selected by a guided random sampling. The procedure continued until there were no more edges to be selected.

Even for large-size instances, heuristics with this type of BRPs can generate a large number of promising solutions in a few seconds, with some of these solutions outperforming the one provided by the original heuristic (Figure 4). Moreover, just by employing different threads the computation can be trivially parallelized by assigning a different seed to each thread, i.e., the BRP allows to generate solutions in real-time that outperform the one provided by the original heuristic. Notice that this might be especially interesting in online optimization problems, where only one or two seconds are allowed before taking a decision –which invalidates the use of time-consuming local search processes. This is the case, for instance, of the mobile cloud computing application analyzed in Mazza *et al* (2016), where mobile users require immediate assignment but, at the same time, some intelligence must be incorporated into the assignment process to optimize the use of shared telecommunication and computing resources. Also, in De Armas *et al* (2016), the authors propose the use of BRPs for fast generation of crew rostering plans in airline companies under realistic conditions. As discussed in Juan *et al* (2014b), another interesting application of these BRPs is the quick generation of a diversified population of starting ‘promising’ solutions for metaheuristics.

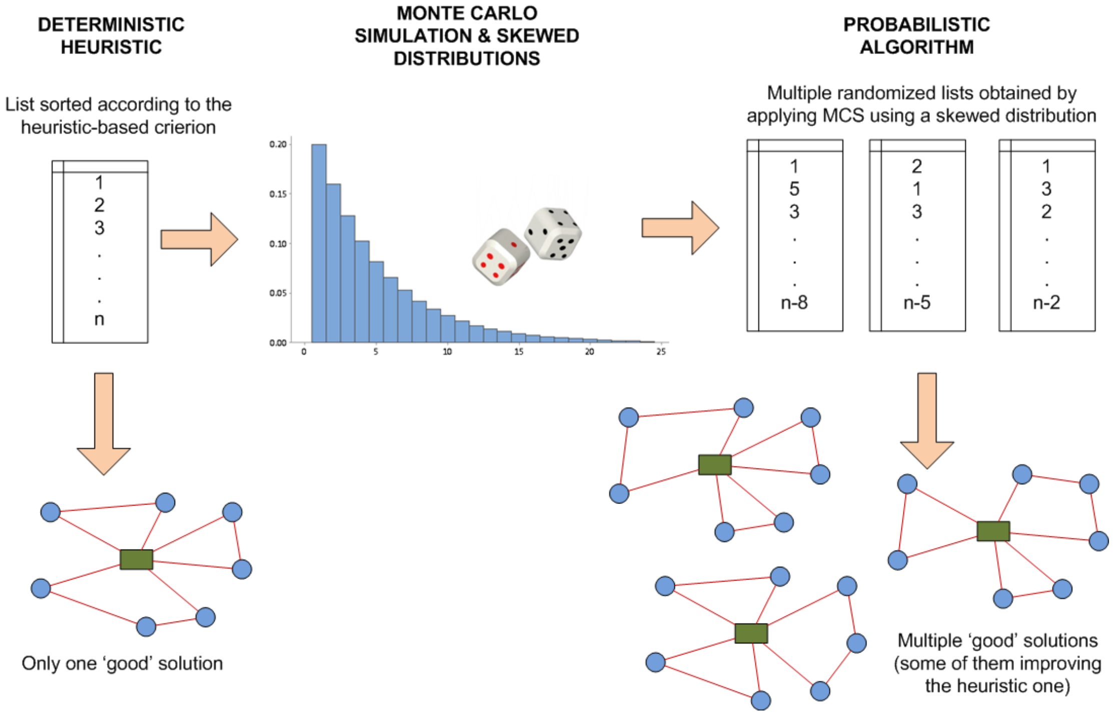
DETERMINISTIC HEURISTIC
MONTE CARLO SIMULATION & SKEWED DISTRIBUTIONS
PROBABILISTIC ALGORITHM
List sorted according to the heuristic-based crierion
1
2
3
n
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
25
Multiple randomized lists obtained by applying MCS using a skewed distribution
1
5
3
n-8
2
1
3
n-5
1
3
2
n-2
Only one 'good' solution
Multiple 'good' solutions (some of them improving the heuristic one)

Figure 4: Using skewed distributions to generate alternative solutions.

From a computational perspective, the generation of random variates coming from probability distributions other than the uniform is always more time-consuming than the generation of pseudo-random numbers. Fortunately, there exist analytical expressions that allow fast generation of random observations from most theoretical distributions. Figure 5 shows an example of Java code that makes use of these analytical expressions to efficiently generate random positions in a sorted list, with these random positions following either a geometric probability distribution or a triangular distribution. Since a large number of such random positions is usually required during the BRP execution, not using such analytical expressions might significantly increase the computational time required by the algorithm. This is, in our opinion, one of the main advantages of using skewed theoretical probability distributions over empirical bias functions, since the latter typically requires more computational time and also more parameter fine-tuning processes than the former.

```
private static int getRandomPosGeom(double beta, Random rng, int n)
{  // Returns random position between 0 and n-1 based on Geometric(beta)
   int index = (int) (Math.log(rng.nextDouble()) / Math.log(1 - beta));
   pos = pos % n;
   return pos;
}


private static int getRandomPosTriang(Random rng, int n)
{  // Returns random position between 0 and n-1 based on decreasing Triangular
   pos = (int) (n * (1 - Math.sqrt(rng.nextDouble())));
   return pos;
}
```

Figure 5: Efficient generation of random positions using skewed distributions.

5. Areas of Application

This section illustrates the use of BRPs with skewed theoretical probability distributions through some examples of applications to different combinatorial optimization problems.

5.1 Vehicle Routing Problem

In a VRP, a set of customers' demands must be satisfied by a fleet of capacitated vehicles that typically depart from a central depot. Moving vehicles between any two nodes (customers or depots) in the map has a distance-based cost. The goal is to find the set of vehicle routes that minimizes the delivery cost while serving all demands and taking into consideration the vehicle capacity constraints. One popular procedure for solving this problem is the aforementioned savings heuristic. In the savings heuristic, an initial dummy solution is constructed by sending a virtual vehicle from the depot to each customer. Then, the list of edges connecting each pair of nodes is considered. This list is sorted according to the 'savings' criterion that would be obtained by using the corresponding edge to merge two routes in the dummy solution. Thus, merging edges associated with higher savings are located at the top of the list, while edges with lower savings are located at its bottom. At this point, the sorted list of edges is traversed from the top to the bottom, and new route merges are carried out whenever the corresponding edge can be used to merge the two routes it connects without violating any constraint.

The diversification of the savings heuristic is rather old, reaching many variants very soon (Toth and Vigo, 2014). As far as we know, the first randomization of the savings heuristic was done by Buxey (1979), who made a random selection of one shortlist of savings according to a probability distribution built taking the savings themselves as weights. Afterwards, Fernández de Córdoba *et al* (1998, 2000) developed two procedures using randomization to solve a real version of the Capacitated VRP and the Rural Postman Problem. Subsequently, the ALGACEA-1 method for the Capacitated VRP included the control of the randomization using an entropy function (Faulin and Juan, 2008).

Nevertheless, as stated above, the first implementations of a BRP with a skewed theoretical distribution was carried out by Juan *et al* (2010). Later, Juan *et al* (2011) improved the algorithm by incorporating some splitting and cache (memory-based) techniques. This conceptual idea was generalized in Juan *et al* (2013a) to the multi-start biased randomization of classical heuristics with adaptive local search (MIRHA). This solution approach was able to handle, in an efficient way, realistic vehicle routing problems under more complex scenarios dominated by non-smooth/non-convex objective functions and non-convex regions.

BRPs of this type were also used in some VRP extensions, namely, the heterogeneous fleet VRP (Juan *et al*, 2014c), the heterogeneous fleet VRP with multi-trips (Caceres-Cruz *et al*, 2014; Grasas *et al*, 2013), the VRP with asymmetric costs and heterogeneous fleets (Herrero *et al*, 2014), the VRP with multiple driving ranges –i.e., heterogeneous fleet with respect to maximum route lengths– (Juan *et al*, 2014d), the multi-depot VRP with a limited number of identical vehicles per depot (Juan *et al*, 2015b), and the two-dimensional loading capacitated VRP with homogeneous fleet (Dominguez *et al*, 2014), with heterogeneous fleet (Dominguez *et al*, 2016b), with heterogeneous fleet and sequential loading and items rotation (Dominguez *et al*, 2016c), and with backhauls (Dominguez *et al*, 2016a).

A similar ‘savings-based’ heuristic, called SHARP, was developed by González *et al* (2012) for solving the arc routing problem. The arc routing problem is similar to the previously described VRP, but it differs in several details: first, demands are not located on the nodes, but on the edges connecting these nodes; second, only some nodes are directly connected among them (i.e., the underlying graph or network connecting the nodes in the problem is not complete). Again, the SHARP heuristic makes use of a dummy initial solution and a sorted list of connecting edges to merge those routes that provide the highest possible savings at each step without violating any problem constraint (e.g., vehicle capacity). The edges are selected with biased probabilities according to a geometric distribution with a parameter randomly selected between 0.10 and 0.25.

5.2 Scheduling Problem

Another well-known optimization problem is the permutation flow-shop problem (PFSP). This is a problem frequently encountered in production processes, where a sequence of jobs or tasks has to be processed in a set of machines. Each job requires a given time to be processed by each machine, and the goal here is to find the permutation of jobs that minimizes the makespan, i.e., the total time necessary to complete the processing of all the jobs in all the machines. The NEH heuristic (Nawaz *et al*, 1983) is probably the best well-known heuristic for solving this problem. In the NEH heuristic, the list of jobs is sorted according to the total time each job would require to be processed by all the machines if it were the only job in the set. Then, the sorted list of jobs is traversed from top to bottom, and a new emerging solution (permutation of jobs) is constructed by locating each new job extracted from the list in the position that minimizes the makespan of the jobs considered so far. Juan *et al* (2014e) employed a BRP with a discretized version of the decreasing triangular distribution during the solution–construction process to select the jobs. This way, eligible jobs were assigned linearly diminishing probabilities according to their corresponding total processing time. In Juan et al (2014a), the former BRP was combined with simulation in order to deal with the PFSP with stochastic processing times.

5.3 Facility Location Problem

The facility location problem (FLP), sometimes referred to as the location-allocation problem, consists of deciding the location of facilities and allocating demand points to one or multiple facilities (Reese, 2006). The objectives can be manifold: minimizing the cost of serving all customers (*p*-median problem), minimizing the longest distance between any customer and its assigned facility (*p*-center problem), minimizing the sum of fixed setup costs and variable costs of serving the customers (uncapacitated facility location problem), minimizing a total cost that is a function of the distance and flow between the facilities plus the fixed cost of placing a facility (quadratic assignment problem), among others.

Cabrera *et al* (2014) modeled a telecommunications problem as an uncapacitated facility location problem, in which web-servers (facilities) needed to be placed in a distributed network to provide some service to a given set of customers. The authors developed a probabilistic algorithm that combined an iterated local search framework with a BRP with a geometric distribution. The use of a biased distribution led to shorter convergence times than those of a uniform distribution.

Similarly, De Armas et *al* (2017) propose a new heuristic for the uncapacitated FLP, and then extend this heuristic to a BRP. They show the efficiency of this approach in solving very large-scale instances in low computing times and, then, they extend the BRP into a simheuristic (Juan et *al* 2015a) able to deal with the stochastic version of the problem.

6. Computational Experiments

In order to provide some empirical evidences on the use of BRP techniques to enhance classical heuristics and quantify the gains obtained, a series of experiments have been developed for five well-known combinatorial optimization problems. The problems, heuristics, and benchmarks selected, as well as the results achieved, are described and analyzed in the next subsections.

6.1 Selected Problems and Heuristics

Five well-known optimization problems have been chosen to illustrate the improvements that can be reached by the introduction of BRPs in constructive heuristics: the VRP, the ARP, the PFSP, the uncapacitated FLP (UFLP), and the 2D strip packing problem (2DSPP).

For the first three problems the following heuristics have been selected: the savings heuristic (Clarke and Wright, 1964) for the VRP, the SHARP heuristic (González *et al*, 2012) for the ARP, and the NEH heuristic (Nawaz *et al*, 1983) for the PFSP. These three heuristics make use of a sorted list that is traversed from the top to the bottom. Thus, at each iteration the next element in the list is chosen without knowing how this selection will condition future decisions during the solution building process. To avoid this greedy behavior, we use a skewed probability distribution to select the next element from the list.

Regarding the UFLP, this is a location problem which involves locating an undetermined number of facilities to minimize the sum of the setup costs of these facilities and the costs of serving the customers from these facilities. It is assumed that there is no limit on the number of customers that can be served from each single facility. In order to solve this problem, we have used the constructive heuristic proposed in De Armas *et al* (2017). This heuristic works as follows: for a given instance, a scenario with all facilities open is considered; then, the marginal savings or loses obtained when each facility is closed in this “all-open” scenario are computed. This way, we obtain a list of possible closures that can be sorted by the savings value. Afterwards, starting from the “all-open” scenario, the savings list is traversed from the beginning, and the next closure is performed as far as it reduces the total cost. After each closure, the savings/losses associated with closing each open facility are updated to take into account the new scenario, and the list is re-sorted accordingly. Obviously, since this heuristic also uses a dynamically-sorted saving list, a BRP technique can be introduced using a skewed probability distribution.

Finally, the 2DSPP –also referred to as the Open Dimension Problem (Wäscher *et al*, 2007)– involves packing items into a single bin or strip of fixed width and infinite height, with the objective of minimizing the total height of the packing within the strip. For this problem we have selected the ‘best-fit decreasing height decreasing width’ heuristic (Mumford–Valenzuela *et al*, 2001; Ntene and Vuuren, 2009). This heuristic is a variation of the ‘first fit decreasing height’ heuristic (Coffman *et al*, 1980). Initially, all rectangles to be packed are sorted by decreasing height (or decreasing width in case of rectangles with equivalent height). Again, a BRP can be applied regarding this sorted list of rectangles to be processed.

6.2 A First Experiment Regarding Parallelization and Computing Times

We have implemented in Java 8 the previously described heuristics and their corresponding multi-start biased-randomized versions for each of the five optimization problems. For all experiments we have used a geometric distribution with a beta parameter adapted to each problem. A series of classical benchmarks were then run on a desktop computer (Intel Core i5 CPU @2.7GHz with 8GB on OS X). Each instance was run 100 times with different seeds as parallel agents, so that each agent is running a certain time. Different time steps have been taken as references to compare the quality of the solutions. In order to compare the heuristic value, *h*, and the best value obtained with the biased-randomized version, *rh*, the percentage gap between both solutions, computed as *gap = (rh – h) / h*, has been used.

More specifically, the benchmark used for the VRP was the classical Kelly instances (Golden *et al*, 1998). This benchmark, available at http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/, is composed of 20 large-scale instances, using from 200 customers to 480. Some instances have restrictions on the maximum length of every route. The benchmark used for the ARP was the classical Egl instances (Li and Eglese, 1996). This set of instances, available at http://logistik.bwl.uni-mainz.de/benchmarks.php, was constructed using as underlying graph regions of the road network of the county of Lancashire, UK. Costs and demands are proportional to the length of the edges, except for non-required edges that have zero demand. The Taillard benchmark is the most used benchmark in the literature for the PFSP (Taillard, 1993). This set of instances, which is available at http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html, is composed of 120 different instances ranging from 20 jobs and 5 machines to 500 jobs and 20 machines. For the UFLP we have selected the Fpp17 benchmark, available at http://www.math.nsc.ru/LBRT/k5/Kochetov/bench.html. It is a set of medium-sized instances introduced by Kochetov and Ivanenko (2003). It consists of 30 instances with 307 customers and 307 facilities. Finally, the Zdf benchmark (Leung and Zhang, 2011; Zhang *et al*, 2013) has been used for the 2DSPP. This benchmark, available at http://paginas.fe.up.pt/~esicup/datasets, is composed of large instances that were generated by combining zero-waste and non-zero waste instances.

Figures 6 to 10 show the gaps for selected instances belonging to each of the benchmarks. The points identify the gaps between the heuristic solution (the highest point) and the different solutions obtained using the randomized version of the heuristic –without any local search added– as the number of parallel agents (executions) and the computing time increases. The lower the point the better the solution and the larger the gap with respect to the original solution provided by the heuristic. For each problem, the graphs clearly show that the quality of the results increases (i.e., the negative gap increases in size) as the number of parallel executions and the total time spent for each of them increase. Therefore, in general, the most promising area of the surfaces corresponds to the corner with the highest number of agents and highest time spent for each agent. Note that the maximum time spent is just a few seconds and there are many cases in which the improvement regarding the original heuristic is between 5% and 10%, so that a big leap in quality is obtained really fast. Being probabilistic algorithms driven by pseudo-random numbers, these biased-randomized algorithms could be easily run in parallel using multiple threads or computers, each of these employing a different seed for the generation of the random variates associated with the different skewed probability distributions. Consequently, investing the same time that the original constructive heuristic (i.e., real-time) it is possible to obtain much better solutions by simply combining BRPs with parallelization and multi-agent strategies, as shown in Juan *et al* (2013b) and Martin *et al* (2016), respectively. Of course, if more time (in the range of seconds) is permitted the quality of the solution improves even further.

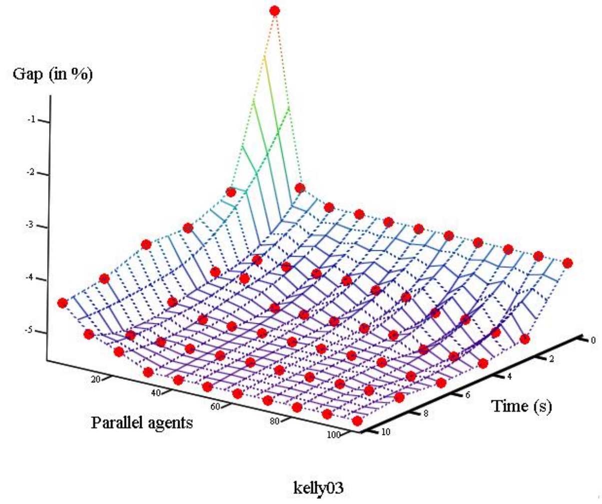

Figure 6: Gap evolution for a VRP instance (Kelly03).

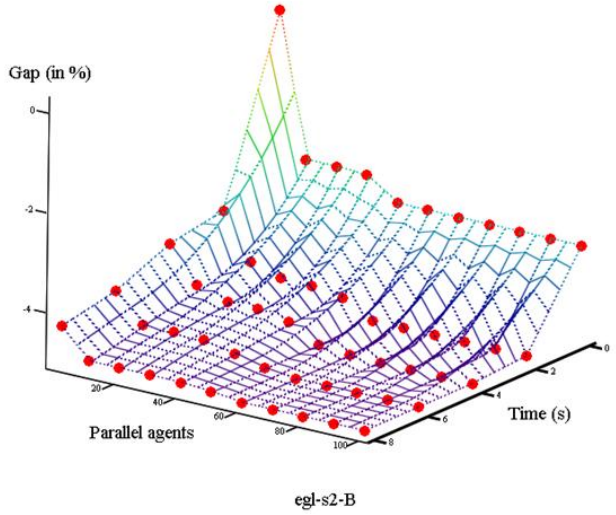

Figure 7: Gap evolution for an ARP instance (egl-s2-B).

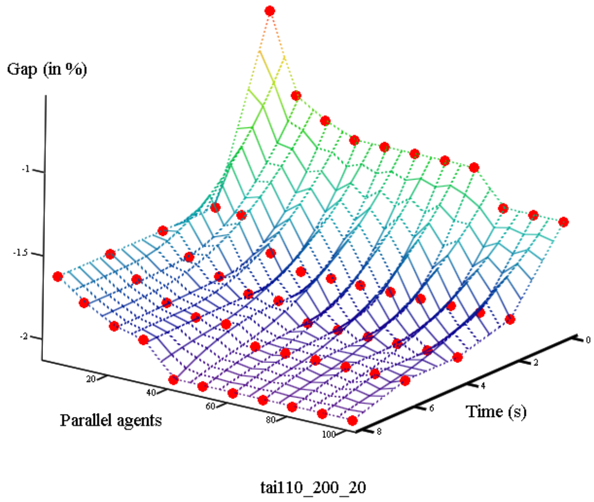

Figure 8: Gap evolution for a PFSP instance (tai110_200_20).

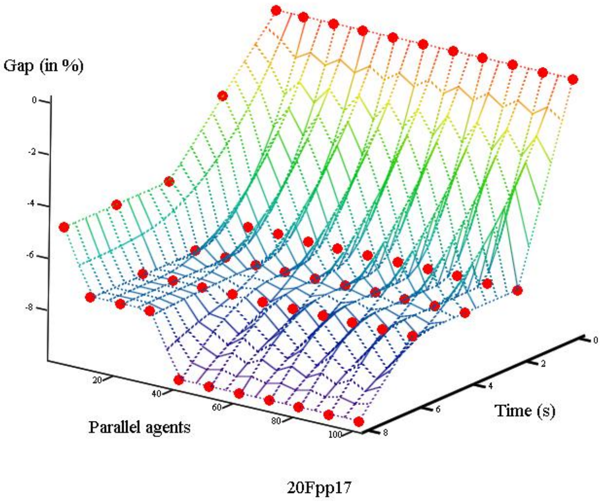

Figure 9: Gap evolution for an UFLP instance (20Fpp17).

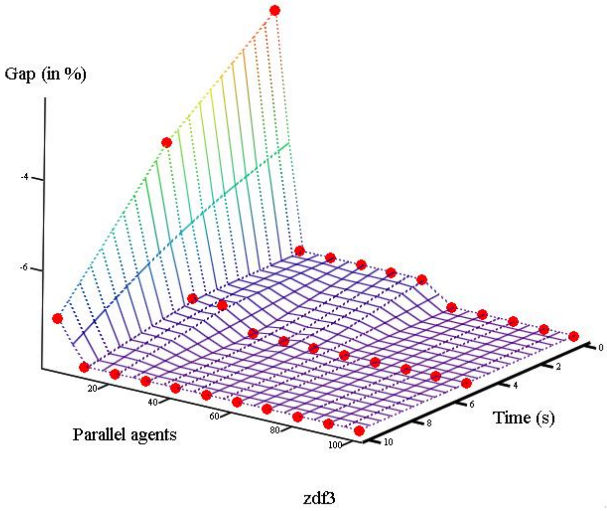

Figure 10: Gap evolution for a 2DSPP instance (zdf3).

6.3 A Second Experiment Comparing BRPs with GRASP-like Approaches

The previous subsection illustrates how biased-randomized and parallelized versions of different heuristics clearly outperform the heuristics themselves in a real-time optimization environment. Here, we compare the performance, also in a real-time optimization environment, of BRPs versus the traditional GRASP randomization process described in Section 2. Thus, for each of the five instances selected (one for each optimization problem considered), both the BRP and the GRASP strategies have been executed using four different parameters (i.e., four different values of the geometric-distribution beta in the case of BRP and four different sizes of the RCL in the case of GRASP). Each of these executions consisted of ten runs (using a different seed for the random number generator at each run), allowing a maximum time of two seconds per run. For each problem, the best value obtained using each approach is shown in Table 1. This table also contains the original value provided by the heuristic as well as the best value obtained after running forty times a uniformly-randomized process (i.e., similar to a GRASP but without restricting the candidate list). The associated gaps of BRP, GRASP, and pure-uniform with respect to the heuristic value are also included (the more negative the gap, the higher the improvement).

Table 1: Comparison of BRPs vs. GRASP randomization in real-time optimization.

Problem	***Heuristic (a)***	***BRP (b)***	***GRASP (c)***	***Uniform (d)***	***Gap (a) - (b)***	***Gap (a) - (c)***	***Gap (a) - (d)***
VRP (kelly03)	12,594	11,718	11,860	91,181	-6.96%	-5.83%	624.00%
ARP (egl-S2-B)	14,124	13,476	13,692	33,118	-4.59%	-3.06%	134.48%
PFSP (tai110_200_20)	11,869	11,644	11,737	11,784	-1.90%	-1.11%	-0.72%
UFLP (20FPP17)	123,245	114,327	114,330	123,349	-7.24%	-7.23%	0.08%
2DSPP (zdf3)	213	197	197	219	-7.51%	-7.51%	2.82%
Averages	--	--	--	--	*-5.64%*	*-4.95%*	*152.13%*

The first thing to be noticed is that, even in real-time, both BRP and GRASP strategies are able to clearly improve the value provided by the original heuristic approach, with negative gaps ranging from -1.11% in the PFSP to the -7.51% in the 2DSPP. The relatively low improvement in the case of the PFSP is probably due to the fact that the NEH heuristic used in this problem employs a simple but efficient local search mechanism. This local search might compensate from ‘bad’ decisions in the order in which jobs are selected -from the list of potential candidates- during the constructive process. Another interesting observation is related to the extremely poor performance of the uniformly-randomization process. As expected, applying a pure-uniform selection process will completely destroy the logic behind the heuristic, thus leading to suboptimal solutions -frequently of lower quality than the one provided by the original heuristic itself-, even for large computational times. Finally, observe that BRP seems to have a superior performance than GRASP, both in average values (-5.64% vs. -4.95%) as well as in the number of significant differences. As shown in Figure 11, BRP clearly outperforms GRASP in three out-of five experiments, while showing a similar behavior in the remaining two (the more external the curve the lower the improvement gap with respect to the solution provided by the associated heuristic).

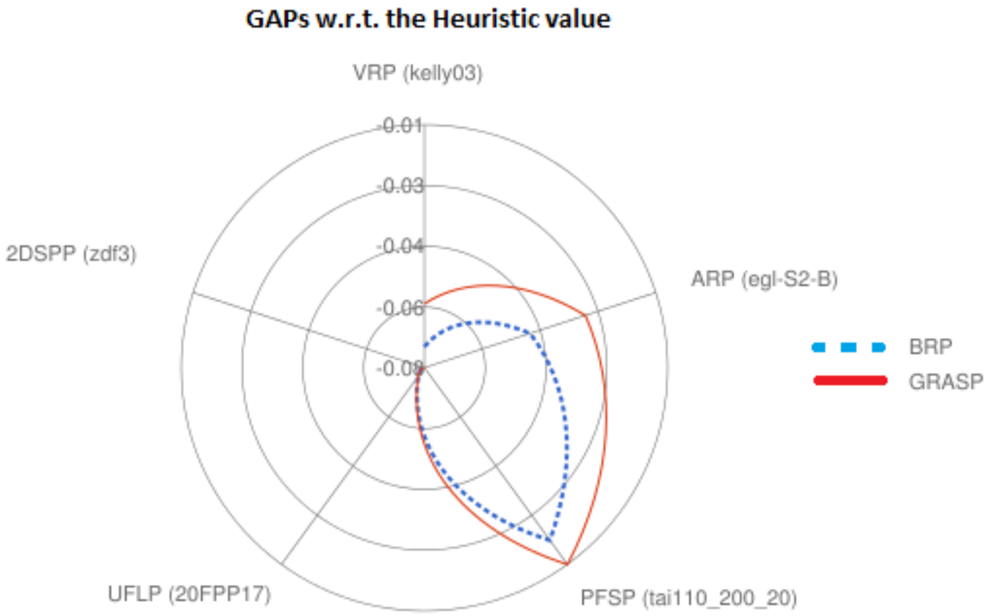

Figure 11: Visual comparison between BRP and GRASP.

7. Conclusions

This work reviews biased randomized procedures (BRPs), their different implementations, and some of their main applications in logistics, transportation, and production. The paper focuses on an emergent family of BRPs that rely on the use of skewed theoretical probability distributions, such as the geometric and the decreasing triangular distributions. These BRPs have two main advantages over more traditional BRPs based on empirical bias functions: *(i)* they are computationally faster, since they benefit from analytical expressions to generate random variates from theoretical probability distributions; and *(ii)* they use at most one parameter that does not require complex and time-consuming setting processes.

By combining skewed probability distributions with random sampling, the logic behind the heuristic can be slightly randomized without losing its good properties. This strategy allows transforming deterministic heuristic procedures into probabilistic algorithms that can be run several times (either sequentially or in parallel) to obtain different promising solutions to the original problem, thus increasing the probability of obtaining better and diversified solutions. As the computational experiments show, the use of BRPs based on skewed probability distributions can easily and noticeably improve the performance of already existing or new heuristics.

Due to their relative simplicity, their fast execution times, and their ability to be parallelized, BRPs constitute an excellent alternative to the use of simple heuristics without incurring in the computational, implementation, and fine-tuning efforts required by most metaheuristics. This is especially the case in online optimization or whenever decisions must be made in real-time even for large-size instances, something that is becoming more frequent due to the growing dynamism, complexity, and responsiveness requirements of most real-life systems in areas such as logistics, transportation, production, telecommunication, finance, Internet computing, health care, etc.

Acknowledgements

TO BE COMPLETED AFTER BLIND PEER-REVIEW PROCESS

References

Aarts E and Lenstra JK (1997). *Local Search in Combinatorial Optimization*. John Wiley & Sons: New York.

Arcus AL (1965). A computer method of sequencing operations for assembly lines. *International Journal of Production Research*, *4*(4), 259–277.

Berry WL, Marcus M and Williams JG (1977). Inventory Investment Analysis Using Biased Sampling Techniques. *Management Science*, *23*(12), 1295–1306.

Bresina J, Drummond M, Swanson K and Edgington W (1994). Automated Management and Scheduling of Remote Automatic Telescopes.In: Pyper DM and Angione RJ (eds).*Optical Astronomy from the Earth and Moon, ASP Conference Series, Vol. 55*.Astronomical Society of the Pacific: San Francisco, pp 216–233.

Bresina JL (1996). Heuristic-Biased Stochastic Sampling. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence- Volume 1*. AAAI Press: Portland, OR, pp 271–278.

Buxey GM (1979). The vehicle scheduling problem and Monte Carlo simulation. *Journal of the Operational Research Society*, *30*(6), 563–573.

Cabrera G, Gonzalez-Martin S, Juan AA, Marquès JM and Grasman SE (2014). Combining Biased Random Sampling with Metaheuristics for the Facility Location Problem in Distributed Computer Systems.In: Tolk A, Diallo SY, Ryzhov IO, Yilmaz L, BuckleyS and Miller JA (eds). *Proceedings of the 2014 Winter Simulation Conference*. IEEE Press: Savannah, GA, pp 3000–3011.

Cáceres-Cruz J, Grasas A, Ramalhinho H and Juan AA (2014). A savings-based randomized heuristic for the heterogeneous fixed fleet vehicle routing problem with multi-trips. *Journal of Applied Operational Research*, *6*(2), 69–81.

Chan Y (2011). *Location Theory and Decision Analysis: Analytics of Spatial Information Technology* (2nd ed.). Springer Berlin Heidelberg: Berlin.

Cicirello VA and Smith SF (2005). Enhancing Stochastic Search Performance by Value-Biased Randomization of Heuristics. *Journal of Heuristics*, *11*(1), 5–34.

Clarke G and Wright J (1964). Scheduling of vehicles from a central depot to a number of delivering points. *Operations Research*, *12*(4), 568–581.

Clerc M (2015). *Guided Randomness in Optimization, Volume 1*. Wiley-ISTE: London.

Coelho J and Tavares L (2003). Comparative analysis of metaheuristics for the resource constrained project scheduling problem. Technical report, Department of Civil Engineering, Instituto Superior Tecnico, Portugal.

Coffman EG, Garey DS and Tarjan RE (1980). Performance bounds for level oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, *9*(4), 808–826.

Collet P and Rennard JP (2006). Stochastic Optimization Algorithms.In: Rennard JP (ed).*Handbook of Research on Nature Inspired Computing for Economics and Management*. Idea Group Inc.: Hershey, PA, pp 28–44.

Cooper DF (1976) Heuristics for Scheduling Resource-Constrained Projects: An Experimental Investigation. *Management Science*, *22*(11), 1186–1194.

Corberán A and Laporte G (2014). *Arc Routing: Problems, Methods, and Applications*. SIAM: Philadelphia, PA.

De Armas J, Cadarso, L, Juan AA and Faulin J (2016). A multi-start randomized heuristic for real-life crew rostering problems in airlines with work-balancing goals. *Annals of Operations Research*, doi:10.1007/s10479-016-2260-y.

De Armas J, Juan AA, Marques JM and Pedroso J (2017). Solving the Deterministic and Stochastic Uncapacitated Facility Location Problem: from a heuristic to a simheuristic. *Journal of the Operational Research Society*, doi: 10.1057/s41274-016-0155-6.

Dominguez O, Guimarans D, Juan AA and Nuez I (2016a). A Biased-Randomised Large Neighbourhood Search for the Two-Dimensional Vehicle Routing Problem with Backhauls. *European Journal of Operational Research*, doi: doi:10.1016/j.ejor.2016.05.002.

Dominguez O, Juan AA, Barrios B, Faulin J and Agustin A (2016b). Using biased randomization for solving the two-dimensional loading vehicle routing problem with heterogeneous fleet. *Annals of Operations Research, 236*(2), 383–404.

Dominguez O, Juan AA, and Faulin J (2014). A biased-randomized algorithm for the two-dimensional vehicle routing problem with and without item rotations. *International Transactions in Operational Research*, *21*(3), 375–398.

Dominguez O, Juan AA, Nuez I De, and Ouelhadj D (2016c). An ILS-Biased Randomization algorithm for the Two-dimensional Loading HFVRP with Sequential Loading and Items Rotation. *Journal of the Operational Research Society*, *67*(1), 37–53.

Dorigo M and Gambardella LM (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, *1*(1), 53–66.

Dorigo M, Maniezzo V and Colorni A (1996). The ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics--Part B*, *26*(1), 29–41.

Drexl A (1991). Scheduling of project networks by job assignment. *Management Science*, *37*(12), 1590– 1602.

Faulin J and Juan AA (2008). The ALGACEA-1 method for the capacitated vehicle routing problem. *International Transactions in Operational Research*, *15*(5), 599–621.

Faulin J, Juan AA, Grasman SE and Fry MJ (2012). *Decision Making in Service Industries: A Practical Approach*. CRC Press: Boca Raton, FL.

Feo T and Resende M (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, *6*(2), 109–133.

Fernández de Córdoba P, García Raffi LM and Sanchis JM (1998). A heuristic algorithm based on Monte Carlo methods for the Rural Postman Problem. *Computers & Operations Research*, *25*(12), 1097–1106.

Fernández de Córdoba P, García-Raffi LM, Mayado A and Sanchis JM (2000). A real delivery problem dealt with Monte Carlo Techniques. *Top*, *8*(1), 57–71.

Giffler B, Thompson GL and Van Ness V (1963). Numerical experience with the linear and Monte Carlo algorithm for solving production scheduling problems. In: Muth JF and Thompson GL (eds). *Industrial Scheduling*. Prentice-Hall, Inc: Englewood Cliffs, NJ.

Glover F (1989). Tabu Search - Part I. *ORSA Journal on Computing*, *1*(3), 190–206.

Glover F (1990). Tabu Search: A Tutorial. *Interfaces*, *20*(4), 74–94.

Golden B, Wasil E, Kelly J and Chao I. (1998). The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: Crainic TG and Laporte G (eds). *Fleet management and logistics*. Springer: New York, pp. 33–56

Gonçalves JF and Resende MGC (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics, 17*(5), 487–525.

González S, Juan AA, Riera D, Castellà Q, Muñoz R and Pérez A (2012). Development and Assessment of the SHARP and RandSHARP algorithms for the Arc Routing Problem. *AI Communications*, *25*(2), 173–189.

Grasas A, Caceres-Cruz J, Lourenço HR, Juan AA and Roca M (2013). Vehicle routing in a Spanish distribution company: Saving using a savings-based heuristic. *OR Insight*, *26*(3), 191–202.

Heller J and Logemann G (1962). An Algorithm for the Construction and Evaluation of Feasible Schedules. *Management Science*, *8*(2), 168–183.

Herrero R, Rodríguez A, Cáceres-Cruz J and Juan AA (2014). Solving vehicle routing problems with asymmetric costs and heterogeneous fleets. *International Journal of Advanced Operations Management*, *6*(1), 58–80.

Juan AA, Barrios B, Vallada E, Riera D and Jorba J (2014a). SIM-ESP: A simheuristic algorithm for solving the permutation flow-shop problem with stochastic processing times. *Simulation Modelling Practice and Theory*, *46*: 101–117.

Juan AA, Cáceres-Cruz J, González-Martín S, Riera D and Barrios BB (2014b). Biased Randomization of Classical Heuristics. In: Wang J (ed). *Encyclopedia of Business Analytics and Optimization, Vol. 1*. IGI Global Books: Hershey, PA, pp. 314–324.

Juan AA, Faulin J, Caceres-Cruz J, Barrios BB and Martinez E (2014c). A successive approximations method for the heterogeneous vehicle routing problem: analysing different fleet configurations. *European J. Industrial Engineering*, *8*(6), 762–788.

Juan AA, Faulin J, Ferrer A, Lourenço HR and Barrios B (2013a). MIRHA: multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problems. *Top*, *21*(1), 109–132.

Juan AA, Faulin J, Grasman S, Rabe M and Figueira G (2015a). A review of Simheuristics: extending metaheuristics to deal with stochastic optimization problems. *Operations Research Perspectives*, *2*: 62–72.

Juan AA, Faulin J, Jorba J, Caceres J and Marques J (2013b). Using Parallel & Distributed Computing for Solving Real-time Vehicle Routing Problems with Stochastic Demands. *Annals of Operations Research*, 207. 43–65

Juan AA, Faulin J, Jorba J, Riera D, Masip D and Barrios B (2011). On the use of Monte Carlo simulation, cache and splitting techniques to improve the Clarke and Wright savings heuristics. *Journal of the Operational Research Society*, *62*(6), 1085–1097.

Juan AA, Faulin J, Ruiz R, Barrios B and Caballé S (2010). The SR-GCWS hybrid algorithm for solving the capacitated vehicle routing problem. *Applied Soft Computing*, *10*(1), 215–224.

Juan AA, Goentzel J and Bektaş T (2014d). Routing fleets with multiple driving ranges: Is it possible to use greener fleet configurations? *Applied Soft Computing*, *21*, 84–94.

Juan AA, Lourenço HR, Mateo M, Luo R and Castellà Q (2014e). Using iterated local search for solving the flow-shop problem: Parallelization, parametrization, and randomization issues. *International Transactions in Operational Research*, *21*(1), 103–126.

Juan AA, Pascual I, Guimarans D and Barrios BB (2015b). Combining biased randomization with iterated local search for solving the multidepot vehicle routing problem. *International Transactions in Operational Research*, *22*(4), 647–667.

King GW (1953). The Monte Carlo Method as a Natural Mode of Expression in Operations Research. *Operations Research*, *1*(2), 46–51.

Kochetov Y and Ivanenko D (2003). Computationally difficult instances for the Uncapacitated Facility Location Problem. In: *Proceedings of the 5th Metaheuristics International Conference (MIC)*: 41:1 – 41:6.

Kolisch R and Hartmann S (1999). Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis.In: Weglarz J (ed).*Project Scheduling: Recent Models, Algorithms and Applications*. Kluwer Academic Publishers: Dordrecht, The Netherlands, pp 147–178.

Lawler EL, Lenstra JK, Rinnooy Kan AHG and Shmoys DB (1985). *The Traveling Salesman Problem*. Wiley & Sons: Chichester.

Leung SCH and Zhang D (2011). A fast layer-based heuristic for non-guillotine strip packing. *Expert Systems with Applications*, *38*(10), 13032–13042.

Li LYO and Eglese RW (1996). An Interactive Algorithm for Vehicle Routeing for Winter-Gritting. *Journal of the Operational Research Society*, *47*(2), 217–228.

Løkketangen A and Glover F (1996). Probabilistic Move Selection in Tabu Search for Zero-One Mixed Integer Programming Problems. In: Osman IH and Kelly JP (eds). *Meta-heuristics: Theory & Applications*. Kluwer Academic Publishers: Boston, MA, pp 467–487.

Mabert VA and Whybark DC (1977). Sampling as a solution methodology. *Decision Sciences*, *8*(1), 167–179.

Martí R, Resende MGC and Ribeiro CC (2013). Multi-start methods for combinatorial optimization. *European Journal of Operational Research*, *226*(1), 1–8.

Martin S, Ouelhadj D, Beullens P, Ozcan E, Juan AA and Burke E (2016). A Multi-Agent Based Cooperative Approach to Scheduling and Routing. *European Journal of Operational Research*, 254(1), 169–178

Mazza D, Pages A, Tarchi D, Juan AA and Corazza G (2016). Supporting Mobile Cloud Computing in Smart Cities via Randomized Algorithms. *IEEE Systems Journal*.

Mumford–Valenzuela C, Wang PY and Vick J (2001). Heuristic for large strip packing problems with guillotine patterns: An empirical study. In: *Proc. 4th Metaheuristics Int. Conference*, pp. 417–421.

Nawaz M, Enscore EE and Ham I (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91–95.

Ntene N and Vuuren JH van (2009). A survey and comparison of guillotine heuristics for the 2D oriented offline strip packing problem, *Discrete Optimization*, *6*(2). 174-188.

Nugent CE, Vollmann TE and Ruml J (1968). An Experimental Comparison of Techniques for the Assignment of Facilities to Locations. *Operations Research*, *16*(1), 150–173.

Pinedo M and Chao X (1999). *Operations scheduling with applications in manufacturing and services*. Irwin/McGraw-Hill: Boston, MA.

Pinedo ML (2012). *Scheduling: Theory, Algorithms, and Systems* (4th ed.). Springer Science & Business Media: New Jersey.

Prais M and Ribeiro CC (2000). Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing, 12*(3), 164–176.

Reese J (2006). Solution methods for the p-median problem: An annotated bibliography. *Networks*, *48*(3), 125–142.

Resende MGC and Ribeiro CC (2010). Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications. In: Gendreau M and Potvin JY (eds). *Handbook of Metaheuristics*. Springer US: New York, pp 283–319.

Schirmer A and Riesenberg S (1997). Parameterized Heuristics for Project Scheduling - Biased Random Sampling Methods. Technical Report 456, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel.

Taillard E (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*(2), 278–285.

Talbi E-G (2009). *Metaheuristics: From Design to Implementation*. Wiley Publishing: New Jersey.

Tonge FM (1965). Assembly Line Balancing Using Probabilistic Combinations of Heuristics. *Management Science*, *11*(7), 727–735.

Toth P and Vigo D (2014). *Vehicle Routing: Problems, Methods, and Applications* (2nd ed.). SIAM: Philadelphia, PA.

Valls V, Quintanilla S and Ballestín F (2003). Resource-constrained project scheduling: A critical activity reordering heuristic. *European Journal of Operational Research*, *149*(2), 282–301.

Wäscher G, Haußner H and Schumann H (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, *183*(3), 1109–1130.

Zhang D, Wei L, Leung SCH and Chen Q (2013). A Binary Search Algorithm base on Randomized local Search for the Rectangular Strip Packing Problem. *INFORMS Journal on Computing*, *25*(2), 332–345.