Heuristic solutions to the Facility Location Problem with General Bernoulli Demands

Maria Albareda-Sambola[(a)], Elena Fernández*[(b)], Francisco Saldanha-da-Gama[(c)]

[(a)]Universitat Politècnica de Catalunya, Dept. d'Estadística i Investigació Operativa, Carrer Colom, 11, 08022 Terrassa, Spain
[(b)]Universitat Politècnica de Catalunya, Dept. d'Estadística i Investigació Operativa, Campus Nord, C5-208, Jordi Girona 1–3, 08034 Barcelona, Spain
[(c)]Universidade de Lisboa, Faculdade de Ciências, Dept. Estatística e Investigação Operacional e Centro de Matemática, Aplicações Fundamentais e Investigação Operacional, Bloco C6, Piso 4, 1749-016 Lisboa, Portugal

February 24, 2017

Abstract

In this paper a heuristic procedure is proposed for the the Facility Location Problem with General Bernoulli demands. This is a discrete facility location problem with stochastic demands that can be formulated as a two-stage stochastic program with recourse. In particular, facility locations and customer assignments must be decided here-and-now, *i.e.*, before knowing the customers who will actually require to be served. In a second stage, service decisions are made according to the actual requests. The heuristic proposed consists of a GRASP followed by a path relinking. The heterogeneous Bernoulli demands make prohibitive the computational effort for evaluating feasible solutions. Thus, the expected cost of a feasible solution is simulated when necessary. The results of extensive computational tests performed for evaluating the quality of the heuristic are reported, showing that high-quality feasible solutions can be obtained for the problem in fairly small computational times.

keywords: discrete facility location, Bernoulli demands, GRASP, path relinking.

1 Introduction

In this paper we propose a heuristic framework for the discrete Facility Location Problem with Bernoulli Demands (FLPBD). The FLPBD consists of a two-stage decision process. In the first stage, facility locations and customer assignments are decided. This must be done *here-and-now*, *i.e.* before knowing the customers who will actually require to be served. Each facility has a capacity in terms of the number of customers it can serve. Moreover, a minimum number of customers must be allocated to every open facility. This threshold is exogenous and location-dependent. In the second stage of the decision process, the customers who actually

*Corresponding author. *e-mail address*: e.fernandez@upc.edu

call for the service are served from their allocated facility, as long as the service capacity is not exceeded. Demand exceeding the service capacity is outsourced and an extra cost is incurred.

The FLPBD aims at modeling situations where a facility provides a service and demand refers to whether a customer requires to be served. Companies providing repair or maintenance services are potential users of the modeling framework we propose. In this case customers can be grouped (e.g. according to their location) and assigned to a facility that should handle the existing demand. The term "facility" should be looked at in a very general way. For instance, we may be referring to a worker or a team. Moreover, facilities may be mobile. One example concerns elevator maintenance: each repair or maintenance team must assist a prespecified set of customers in case they call for service. If the actual demand turns out to be higher than the service capacity, then the service still has to be provided, which may call for temporary relocation of workers from other teams or simply for outsourcing the service to a third party. Another potential application of the FLPBD concerns mobile health clinics. This type of facility is usually set to assist some specific area or region previously assigned to it. In case the occurring demand is higher than the service capacity, extra personnel is necessary, which may lead to additional costs. Other examples of settings fitting the FLPBD include target-oriented advertisement activities, door-to-door product demonstration, etc. In all these cases, potential customers are previously assigned to the facility and may or may not have actual demand.

Although stochastic facility location problems have received much less attention than their deterministic counterparts, a variety of models and approaches have been studied in this context (Louveaux 1993; Snyder 2006; Correia and Saldanha-da-Gama 2015). In particular, the FLPBD was introduced by Albareda-Sambola et al. (2011) together with two different outsourcing policies. Although the problem is presented in its general version, that paper concentrates mainly on the particular case where all customers share the same demand probability, which is referred to as homogeneous case. For this case closed forms for the recourse functions associated with the two outsourcing policies are presented, and compact formulations of the deterministic equivalent problem are developed, which allow solving the problem exactly for moderate instance sizes. Nevertheless, the methodology proposed by the authors becomes computationally demanding as the sizes of the instances increase and it is only valid for the homogeneus case. More recently, Bieniek (2015) showed that the expressions of the recourse function can be extended to some distributions different from the Bernoulli, as long as the assumption of homogeneity among customers is kept.

In this work we focus on the general version of problem, *i.e.*, the demand probabilities are not necessarily equal. We consider the two outsourcing strategies studied by Albareda-Sambola et al. (2011). To the best of the authors' knowledge no methodology has been proposed so far for this problem. The only exception is the companion paper Albareda-Sambola et al. (2016), where different outsourcing policies are analyzed and compared for the situation where the uncertainty is captured by a moderate number of scenarios. As shown in that work, MIP formulations can be used within a sample average approximation algorithm to tackle the FLPBD. Nevertheless, this renders a rather inefficient approach, that easily becomes unaffordable even for moderate-sized instances. We observe that in the general FLPBD computing the exact cost of a particular solution is already computationally unaffordable since it requires computing the expected value of a random variable by enumerating all its possible values. Therefore, at all phases of the heuristic we resort to simulating these costs.

In order to obtain high-quality feasible solutions we have designed a heuristic based on GRASP and path relinking (PR). PR algorithms focus on exploring the feasible region by

following trajectories connecting pairs of different feasible solutions (Glover 1997, Glover et al. 2000) and have been successfully combined with GRASP for many combinatorial optimization problems using different strategies (Festa and Resende 2013). In our particular case, we combine them within a two-phase algorithm. The first phase applies a GRASP for building two pools of feasible solutions, one focusing on quality and the other focusing on diversity. For the construction phase of the GRASP some structural constraints are temporarily relaxed. Therefore, a feasibility restoration mechanism may be required. Additionally, a local search is applied for improving the solution. Once the two pools have been built, the second phase of the heuristic starts, which makes use of the actual PR procedure. The idea is to consider repeatedly one solution selected from the quality pool and another one selected from the diverse pool and then explore a "path" linking these solutions in an attempt to find better feasible solutions. The process continues until some stopping criteria are met.

Many authors have combined intensification and diversification strategies in their heuristics (Vidal et al. 2013; Cataruzza et al. 2014). In our case this is accomplished by building the two different solution pools in the GRASP phase. Taking one solution from each pool in each iteration of the PR phase gives the algorithm a large degree of flexibility, which allows overcoming the inefficiencies that GRASP alone might have for large instances.

The results of extensive computational tests performed using different types of instances are reported. These results show that within very small computational times the new heuristic finds high-quality solutions for the FLPBD with heterogeneous demand probabilities as well as for the particular case of homogeneous demand, where our solutions can be contrasted against the optimal ones.

The remainder of this paper is organized as follows. In Section 2, we formally define the FLPBD. In Section 3 we introduce the new heuristic and in Section 4 we report and discuss the computational experience performed. The paper finishes with Section 5 where we summarize the main findings of this work.

2 Problem definition

Let I and J (with $m = |I|$ and $n = |J|$) denote the sets of indices for the potential locations of facilities and for the customers, respectively. We assume that the demand for service of each customer $j \in J$ is given by a binary random variable that we denote by ξ_j, indicating whether or not customer j requests the service. Additionally, we assume that ξ_j variables follow independent Bernoulli probability distributions with parameters p_j, $j \in J$. Furthermore, we impose that if a facility is opened then the number of customers assigned to it must be above some threshold. Finally, we assume a limited service capacity in each location.

Imposing to assign a minimum number of customers to each opened facility is a way of guaranteeing a (potential) minimum workload for the open facilities. Additionally, looking into the literature, we can find an increasing number of applications (e.g., in logistics systems design), calling for such minimum thresholds at the stage of dimensioning facilities (see, e.g., Melo et al. 2006, 2009). We note that when such constraints are not necessary, we can simply set the minimum threshold to 0 and the proposed methodology is still fully valid.

For $i \in I$, we consider the following notation:

f_i, fixed setup cost for opening facility i;
ℓ_i, minimum number of customers that have to be assigned to facility i if it is opened;
K_i, maximum number of customers that can be served from facility i if it is opened;
c_{ij}, cost for serving customer j from facility i ($j \in J$).

In a particular realization of the random vector there may exist customers who do not request the service. Hence, we distinguish between the *assignment* of customers to the facilities—which is done *a priori* and is independent of the potential realizations—and the *service* provided to customers from the open facilities—which is decided *a posteriori*, once the realization of the random vector is known. Throughout the text customers calling for service in a particular realization of the uncertainty are referred to as *demand customers*. We consider single assignment, *i.e.*, each customer is assigned to one and only one open facility.

An *a priori* solution for the FLPBD is defined by a set of operating facilities together with an assignment of all the customers to these facilities. It should be noted that the service capacity of the facilities (K_i) does not affect the feasibility of *a priori* solutions.

For an *a priori* solution we denote by J_i (with $z_i = |J_i|$) the set of customers assigned to facility i and by η_i the random variable counting the number of demand customers in J_i. An *a posteriori* solution specifies the decisions to make once demand customers are known, *i.e.*, it explicitly states how demand customers are served. If $z_i \leq K_i$ for some $i \in I$ then in the *a posteriori* solution all the demand customers indexed in J_i receive service from facility i, each of them incurring a service cost c_{ij}, $j \in J_i$. However, if $z_i > K_i$ then we may also observe $\eta_i > K_i$. In this case, the *a posteriori* solution serves K_i out of the η_i demand customers from plant i and outsources the service of the remaining ones. We assume that the requests for service arrive before the second stage decision is made. Therefore, the arrival order of service requests can be used as a criterion to prioritize customers when some outsourcing is needed. A cost g_i is incurred for every outsourced demand customer.

The FLPBD is to find a set of facilities to open and an allocation of the customers to those facilities satisfying the lower bounds ℓ_i. The goal is to minimize the sum of the fixed costs associated with the open facilities plus the recourse function. Albareda-Sambola et al. (2011) introduced the following sets of decision variables in order to present a formulation for this problem: y_i is a binary variable equal to 1 if facility $i \in I$ is opened and 0 otherwise; x_{ij} is a binary variable equal to 1 if customer $j \in J$ is allocated to $i \in I$ and 0 otherwise. The FLPBD can be formulated as follows:

$$(P)\ \text{minimize} \quad \sum_{i \in I} f_i y_i + \mathcal{Q}(x), \tag{1}$$

$$\text{subject to} \quad \sum_{i \in I} x_{ij} = 1, \qquad j \in J, \tag{2}$$

$$x_{ij} \leq y_i, \qquad i \in I,\ j \in J, \tag{3}$$

$$\ell_i y_i \leq \sum_{j \in J} x_{ij}, \quad i \in I,\ j \in J, \tag{4}$$

$$y_i \in \{0,1\}, \qquad i \in I, \tag{5}$$

$$x_{ij} \in \{0,1\}, \qquad i \in I,\ j \in J. \tag{6}$$

The objective function (1) includes the fixed costs for opening the facilities plus the recourse function. The latter can be decomposed according to $\mathcal{Q}(x) = \mathbb{E}_{\xi,\eta}(\text{Service cost} +$

Penalty cost). Constraints (2) ensure that all customers are assigned to (exactly) one facility while constraints (3) impose that these assignments are only made to operating facilities. Constraints (4) state the minimum number of customers that must be assigned to each operating facility. Finally, (5) and (6) define the domain of the decision variables.

2.1 Outsourcing policies

Albareda-Sambola et al. (2011) proposed two outsourcing strategies (recourse actions) within the context of the FLPBD that we also consider in this paper.

Facility outsourcing.

Under this strategy an open facility $i \in I$ serves all the demand customers assigned to it. However, when $\eta_i > K_i$ the facility acquires the extra resources needed for serving $\eta_i - K_i$ customers—the number of outsourced customers—at a unitary cost g_i. Then the facility serves each demand customer $j \in J_i$ assigned to it at the service cost c_{ij}. Therefore, the recourse cost in this case does not depend on who are the $\eta_i - K_i$ outsourced customers.

This recourse action models situations where a temporary expansion of the serving capacity of the facilities is possible.

Customer outsourcing.

Under this strategy when the number of demand customers assigned to facility i exceeds the service capacity, K_i exactly K_i customers are served from the facility, each of them at the corresponding service cost c_{ij}. The remaining $\eta_i - K_i$ customers are served directly from an external service provider at a unitary cost g_i.

We assume that the customers to be served from the facility are selected according to a FIFO policy, *i.e.*, according to the order in which service requests have arrived. We further assume that service requests arrive in a random order and every sorting can occur with the same probability.

Similarly to Albareda-Sambola et al. (2011) we consider unitary outsourcing costs g_i, $i \in I$, which are facility dependent. This is motivated by the fact that when the service capacity is exceeded, the facility providing the service remains responsible for assuring that the customers are served. This holds, for instance, at the expenses of extra workforce—facility outsourcing—or by contracting the service from a third party—customer outsourcing.

As we have already mentioned, due to the heterogeneous demand probabilities, evaluating feasible solutions is computationally unaffordable for both outsourcing policies. This is particularly true for customer outsourcing since the order by which the customers call for service is also needed to define a particular realization of the uncertain data.

3 A GRASP with path relinking for the FLPBD

GRASP (Greedy Randomized Adaptive Search Procedure) is a well-known multi-start metaheuristic (Feo and Resende 1995). Each iteration consists of two phases: (i) construction, and (ii) local search. The construction phase aims at building a feasible solution; the local search attempts to improve it.

GRASP has been successfully used in the context of many complex optimization problems and, in particular, deterministic discrete location problems (see, for instance, Resende and Werneck 2006, Delmaire et al. 1999, and Díaz and Fernández 2006). It has also been applied successfully to different types of stochastic problems such as those investigated by Ballestín and Leus (2009) and Held and Woodruff (2005). The interested reader is referred to Resende and Ribeiro (2008) and Resende and Ribeiro (2013) for comprehensive surveys on this methodology.

Path relinking was originally proposed by Glover (1997) as an intensification scheme. It explores trajectories connecting pairs of feasible solutions found by metaheuristics such as tabu search and scatter search (Glover and Laguna 1997). The original idea consists of taking two elite solutions at a time—a leading solution and a target solution—and then progressively changing the leading solution in order to reach the target by repeatedly performing local moves. In this process, better feasible solutions may be found due to the combination of good attributes of the target and leading solutions.

The use of path relinking combined with GRASP was first proposed by Laguna and Martí (1999) and led to many other extensions and successful applications to well-known discrete combinatorial optimization problems (see Resende and Ribeiro 2013, as well as the references therein). Several strategies are possible. Two popular ones are the following: (i) path relinking is applied to all pairs of elite solutions (during the GRASP or after all GRASP iterations are performed); (ii) path relinking is used as an intensification strategy taking a locally optimal solution produced by each GRASP iteration after local search, and an elite solution.

AKI

In this work we propose a heuristic for the FLPBD that consists of a GRASP followed by a PR. The GRASP aims at building two pools of solutions. One—the so-called *elite pool*—focuses on quality and contains a certain number of the best solutions found during the execution of the GRASP. The other one—*diverse pool*—focuses on diversity and consists of solutions that were not good enough to be part of the elite pool but that are the most diverse ones among the solutions currently in both pools. Both pools have a limited size that we denote *nElite* and *nDiverse* for the *elite pool* and the *diverse pool*, respectively. When one pool is full and we want to insert one solution there, then the worst solution in the pool is removed.

When both pools have been built, a PR procedure starts. It repeatedly chooses a target solution from the *elite pool* and then, starting from a diverse solution selected at random, explores a path linking them in an attempt to find better feasible solutions.

The overall procedure is summarized in Algorithm 1, where we can clearly observe the two above mentioned phases. In the first phase (lines 1 to 17) the GRASP is executed max_iter times. In each execution, a solution S is constructed (line 5—function `GreedyRandomizedConstruction(`α`,`p^+`)`) and repaired (`RepairSolution(`S`)`) if it is infeasible (lines 6 and 7). A local search (`LocalSearch_1(`S`)`) is applied at the end. All moves are evaluated by estimating their impact in the solution cost.

Once a solution is built, we decide whether to insert it in one of the pools—lines 9 to 16. The actual value of the feasible solution resulting from the local search is estimated (`estimateCost(`S`,`$highPrecision$`)`) and we check if its quality indicates that it should be inserted into the *elite pool*. If this is the case we insert the solution in the pool (`insertInElitePool(`S`,`$worstValueElite$`)`), updating (when necessary) the worst value among all solutions of the pool. Otherwise, we check whether the solution should be inserted into the *diverse pool*. In this case the solution enters that pool (function `insertInDiversePool(`S`,`

Algorithm 1 Heuristic framework for the FLPBD.

```
    // initialization
 1: elite_pool ← ∅;   worstValueElite ← ∞;
 2: diverse_pool ← ∅;   worstDiverseValue ← 0.0;
 3: initializeConstruction()
    // phase 1: GRASP used for building two pools of solutions.
 4: for k = 1, ..., max_iter do
 5:     S ← GreedyRandomizedConstruction(α, p+);
 6:     if S not feasible then
 7:         S ← RepairSolution(S);
 8:     S ← LocalSearch_1(S)   // cost variation estimated in all moves
 9:     f(S) ← estimateCost(S, highPrecision)   // solution cost estimated at the end
                                                   of the local search
10:     if S ∉ elite_pool and f(S) < worstValueElite then
11:         elite_pool ← insertInElitePool(S, worstValueElite);
12:     else
13:         if S ∉ diverse_pool then
14:             dist(S) ← pool_distance(S);
15:             if dist(S) > worstDiverseValue then
16:                 diverse_pool ← insertInDiversePool(S, worstDiverseValue);
17: S_incumbent ← arg min_{S∈elite_pool}{f(S)};   // best solution so far
    // end of phase 1.
    // phase 2: a PR procedure is used for improving the solution.
18: for ℓ = 1, ..., nRep do
19:     for all S ∈ elite_pool do
20:         S_target ← S;
21:         S_current ← getRandom(diverse_pool);
22:         S_incumbent ← pathRelinking(S_current, S_target, S_incumbent);
23: return S_incumbent.
    // end of phase 2.
```

worstDiverseValue)) and the worst diverse value among solutions in the pool is updated (if necessary).

The implementation details of the function estimating the cost of a solution (`estimateCost(`S`,` *highPrecision*`)`) are provided in Section 4.2.

At the end of the GRASP (line 17) we set as the incumbent solution the best one found in all executions of the GRASP. This solution is the one in the *elite pool* with the smallest estimated cost. Then we start the PR procedure (lines 18–23). One path is explored for pairs of feasible solutions such that one is selected from the *elite pool* and the other one is selected

from the *diverse pool*. The former is selected sequentially and is set as the target solution; the latter is taken randomly and is set as the current solution. After defining a target and a current solution, a path linking them is explored. The incumbent solution is updated every time a solution is found with a better cost estimate. As we can observe in Algorithm 1, the PR procedure is repeated $|elite\_pool| \times nRep$ times.

In the next subsections we detail the different functions invoked by this heuristic.

3.1 The greedy randomized procedure

The construction phase of a GRASP is the randomization of a greedy algorithm. At each iteration the set of candidate elements consists of all components that can be incorporated into the solution under construction and satisfy some pre-specified criterion. All candidate elements are evaluated according to a greedy function and a restricted candidate list (RCL) is created considering the best elements relative that function. The element to be incorporated into the solution under construction is randomly selected from the RCL.

At each iteration of the constructive phase we have a partial solution (a subset of open facilities) and a set of candidate elements (facilities that are not open). Furthermore, our constructive phase ignores constraints (4) concerning the minimum number of customers assigned to open facilities. Accordingly, the outcome of this phase may be infeasible. In these situations a feasibility restoration mechanism is applied. Finally, a local search procedure is employed in an attempt to improve the feasible solution obtained. All these elements are detailed next.

Observe first that the construction phase uses auxiliary assignment capacities for all facilities, which remain constant for all executions of the GRASP. We set them proportional to the corresponding service capacities, but large enough to allow the allocation of all customers to the facilities. In particular, for each $i \in I$ we define such capacities according to

$$u_i = \max \left\{ \ell_i, K_i, \left\lfloor \frac{n}{\overline{p}} \frac{K_i}{\sum_{t \in I} K_t} \right\rfloor \right\},$$

where $\overline{p}$ denotes the average of the demand probabilities p_j, $j \in J$.

3.1.1 Construction phase

This phase starts by choosing one single facility to open and then assigning all customers to it. The facility is chosen randomly from the RCL. In a general step k, a new facility is opened and some customers are reassigned to it, thus producing a new partial solution $(\tilde{\mathbf{y}}^{\mathbf{k}}, \tilde{\mathbf{x}}^{\mathbf{k}})$. At the end of each step k, we denote by I^k the set of open facilities and by $i(j)$ the plant customer $j \in J$ is currently assigned to. The full procedure is detailed in Algorithm 2.

Typically, the greedy evaluation function used in the constructive phase of a GRASP estimates the increase in the objective function due to the incorporation of an element into the partial solution under construction. In our case we evaluate differently the incremental costs of the candidates in the first and the subsequent iterations. In the first iteration each

candidate is evaluated according to

$$\delta_i^0 \leftarrow \frac{1}{u_i}\left(f_i + \sum_{j\in J} c_{ij}\right). \tag{7}$$

This initialization is made in line 2. Instead, in subsequent iterations, for a candidate facility i we estimate the incremental cost δ_i^k (line 18) as the sum of the setup cost of the facility plus an estimate of the variation in the service costs when reassigning customers from their current assignments to the candidate facility i, including the expected saving in the penalty for unserved customers as well as K_i. For each customer $j \in J$ the estimated variation in its service cost is computed as (line 15)

$$\sigma_{ij} = c_{ij} - c_{i(j)j} - p_j \times g(i(j))\frac{(z_{i(j)} - K_{i(j)})^+}{z_{i(j)}}. \tag{8}$$

We recall that z_i denotes the number of customers assigned to facility $i \in I$. The actual number of customers that would be assigned to facility i if it is open, r_i, is determined in line 17, taking into account the upper bound u_i, but aiming at satisfying the lower bound ℓ_i and also trying to reallocate as many customers as possible, among the ones with negative estimate of their reassignment costs. Then, in line 21 the RCL is built with the non-open facilities that seem most promising if opened.

The criterion we apply for defining the elements of the RCL also changes from the first to subsequent iterations. In all cases the RCL contains all closed facilities with an incremental cost within the interval $[\delta^{\min}, \delta^{k\min} + \alpha^k(\delta^{k\max} - \delta^{k\min})]$, where, for each iteration k, $\delta^{k\min}$ and $\delta^{k\max}$ denote, respectively, the smallest and largest non-positive incremental costs as defined above. In the initial iteration ($k = 0$) $\alpha^k = 2\alpha$, where α is a parameter to be defined, whereas in subsequent iterations ($k > 0$) $\alpha^k = \alpha$. The next facility to open is randomly chosen from the RCL.

Preliminary tests showed that the choice of the first facility to open has a great influence on the resulting solution. For this reason, with the goal of generating diverse solutions, we use a wider RCL in the first step ($\alpha^0 = 2\alpha$). More sophisticated techniques can be used to auto-tune the parameter α^k. One such possibility is the so-called reactive GRASP considered by Delmaire et al. (1999) and Ríos-Mercado and Fernández (2009). In that case the range of the RCL is adjusted during the GRASP phase. We did not attempt such alternative since high quality solutions could be obtained for the FLPBD by the procedure we propose considering a fixed RCL range.

Usually the constructive phase would continue until all facilities are open or until all the non-open facilities have a non-negative incremental cost. Preliminary computational testing, however, indicated that these criteria tend to produce solutions with too many open facilities. For this reason we have introduced one additional parameter, p^+, which denotes the probability with which we continue the process even if none of the above termination criteria are met (lines 10, 11).

Algorithm 2 `GreedyRandomizedConstruction`(α,p^+, u)

// choose the first facility to open
$k \leftarrow 0$
Compute $\{\delta_i^k\}_{i \in I}$ using (7).
RCL $\leftarrow \left\{ i \in I : \delta^{k\min} \leqslant \delta_i^k \leqslant \delta^{k\min} + 2\alpha(\delta^{k\max} - \delta^{k\min}) \right\}$;
$i^k \leftarrow$ `RandomSelect(RCL)`;
$I^k \leftarrow \{i^k\}$
$i(j) \leftarrow i^k$, $j \in J$;
$z_{i^k} \leftarrow n$, $z_i \leftarrow 0$, $i \in I \setminus \{i^k\}$;
// main loop
repeat
    $k \leftarrow k + 1$
    $\beta \leftarrow$ `RandomSelect([0,1])`
    if $(\beta > p^+)$ **then** Stop.
    else
        for $(i \in I)$ **do**
            for $(j \in J)$ **do**
                compute σ_{ij} according to (8)
            $\{j_{[1]}, \ldots, j_{[n]}\} \leftarrow$ `sort_σ_increasing(J)`
            $r_i \leftarrow \min\left\{u_i, \max\left\{\ell_i, \max\{q : \sigma_{ij_{[q]}} < 0\}\right\}\right\}$
            $\delta_i^k \leftarrow f_i + \sum_{t=1}^{r_i} \sigma_{ij_{[t]}}$
        $\delta^{k\min} \leftarrow \min\left\{0, \min\{\delta_i^k : i \in I\}\right\}$; $\delta^{k\max} \leftarrow \min\left\{0, \max\{\delta_i^k : i \in I\}\right\}$
        if $\delta^{k\min} < 0$ **then**
            RCL $\leftarrow \left\{ i \in I : \delta^{k\min} \leqslant \delta_i^k \leqslant \delta^{k\min} + \alpha(\delta^{k\max} - \delta^{k\min}) \right\}$
            if RCL $\neq \emptyset$ **then**
                $i^k \leftarrow$ `RandomSelect(RCL)`
                $I^k \leftarrow I^{k-1} \cup \{i^k\}$
                for $(j = [1], \ldots, [r_{i^k}])$ **do**
                    $z_{i(j)} \leftarrow z_{i(j)} - 1$; $i(j) \leftarrow i^k$
                $z_{i^k} \leftarrow r_{i^k}$
until $\delta^{k\min} \geq 0$
return I^k and $i(j)$, $j \in J$

3.1.2 Feasibility restoration

When the outcome of the construction phase violates some of the lower bounds ℓ_i for the assignments to the open facilities, a feasibility restoration procedure is required (Algorithm 1—

function `RepairSolution(`S`)`).

The feasibility restoration mechanism that we propose completely redefines (if necessary) the assignments of customers to the plants opened in the constructive phase. This is accomplished in two steps:

1. We check whether $\sum_{i \in I^k} \ell_i > n$ for the constructed solution. If this is the case then at least one facility must be closed. Accordingly, we start closing facilities (one each time) until $\sum_{i \in I^k} \ell_i \leq n$. In each iteration, we choose for closing a facility i^* such that

$$i^* \in \arg \max_{i \in I^k \,|\, (\ell_i - z_i) > 0} \left\{ f(i) + \sum_{j \in J \,|\, i(j) = i} c_{ij} \quad + g(i) \left(\overline{p}_i \times z_i - K_i \right)^+ \right\},$$

   where $\overline{p}_i$ denotes the average demand probability of all customers assigned to facility i.

   When a facility is closed all the customers who are assigned to it must be reassigned to the remaining facilities. This is done in a greedy way, *i.e.*, the reassignment chosen is the one that contributes the least to the (estimated) increase in the solution cost. We start by considering reassignments only to the facilities that remain infeasible. When such facilities no longer exist, we consider reassignments to the other facilities. We estimate the cost of reassigning some customer j to a facility i as follows:

$$c_{ij} + \max\{z_i + 1 - K_i, 0\} p_j \frac{g_i \times K_i}{z_i \times (z_i + 1)}.$$

2. By ensuring that $\sum_{i \in I^k} \ell_i \leq n$ (which is done in the previous step) we know that with the currently open facilities constraints (4) can be satisfied. However, the solution is not necessarily feasible even when this condition is met. Indeed, for some facility i we may still have $z_i < \ell_i$. In this case we reassign customers who are currently assigned to facilities i' such that $z_{i'} > \ell_{i'}$ to facility i. Again this is done in a greedy way: the reassignment chosen is the one that increases the least the solution cost. The reassignment cost is estimated according to

$$p_j(c_{ij} - c_{i'j}) + \max\{z_i + 1 - K_i, 0\} p_j \frac{g_i \times K_i}{z_i \times (z_i + 1)} - \max\{z_{i'} - K_{i'}, 0\} p_j \frac{g_{i'} \times K_{i'}}{z_{i'} \times (z_{i'} - 1)}.$$

3.1.3 Local search

The local search attempts to iteratively improve the solutions obtained in the previous phases. This is accomplished by successively replacing the current solution by a better one from its neighborhood, according to its estimated cost. The search terminates when no such solution can be found so that the current solution seems to be locally optimal.

The neighborhoods that we considered for the local search in our GRASP do not change the set of open facilities; they only affect the assignment of customers within that set. In particular, we consider two different neighborhoods: (i) those induced by *reassignments* of customers within the set of open facilities, and (ii) those induced by *interchanges* of assignments between pairs of customers.

In both cases we apply a first improving policy relative to the estimated variations in the solution cost. They are computed as follows:

Reassignments

The variation in cost when reassigning customer j from plant $i_1 = i(j)$ to i_2 can be approximated by:

$$\sigma_{j,i_2} = p_j(c_{i_2 j} - c_{i_1 j}) + \max\{z_{i_2} + 1 - K_{i_2}, 0\} g_{i_2} p_j \times \sum_{s=K_{i_2}}^{z_{i_2}} \text{bin}(z_{i_2}, \bar{p}_{i_2}, s)$$

$$- \max\{z_{i_1} - K_{i_1}, 0\} g_{i_1} p_j \times \sum_{s=K_{i_1}+1}^{z_{i_1}} \text{bin}(z_{i_1}, \bar{p}_{i_1}, s),$$

where $\overline{p}_{i_1}$ and $\overline{p}_{i_2}$ represent the arithmetic average of the demand probabilities for the customers currently assigned to i_1 and i_2, respectively, and $\text{bin}(z, p, s)$ stands for the probability that a binomial random variable with parameters z and p takes value s; *i.e.* $\text{bin}(z, p, s) = \binom{z}{s} p^s (1-p)^{z-s}$.

In the above expression, we combine three estimates: (i) the increase of the service costs, (ii) the increase of the expected penalty in plant i_2, which will have one more customer now, and (iii) the reduction of the expected penalty in plant i_1. Feasibility is kept by considering only customers j with $z_{i(j)} > \ell_{i(j)}$.

Since cost variations are estimated, we may end up with a cycle of reassignments. In order to avoid this we estimate the variation in the cost associated with the reverse move before reassigning a customer. We only perform the reassignment if this estimate is larger than the estimated variation for the direct move.

Interchanges

We consider the interchange in the assignment of two customers j_1, j_2, with $i_1 = i(j_1), i_2 = i(j_2)$, where, without loss of generality, we assume that $p_{j_1} > p_{j_2}$. Our estimate of the cost variation is:

$$\sigma_{j_1 j_2} = c_{i_1 j_2} + c_{i_2 j_1} - c_{i_1 j_1} - c_{i_2 j_2} + g_{i_2} (p_{j_1} - p_{j_2}) \left[\frac{(z_{i_1} - K_{i_1})^+}{z_{i_1}} - \frac{(z_{i_2} - K_{i_2})^+}{z_{i_2}} \right].$$

Both for the reassignments and for the interchanges, we set a maximum number of times the same assignment of a customer to a facility can be made. We alternate the exploration of the two neighborhoods until no promising moves are identified.

3.1.4 Diversity measure

In the GRASP phase, when a solution S is not inserted in the *elite pool* we try to insert it into the *diverse pool*. In order to do so we measure the diversity of S (`pool_distance(`S`)`). This is done by comparing S with every solution in one of the pools and counting the number of different customer assignments. In particular, we compute

$$\frac{1}{n_e + n_d} \left(\sum_{S_e \in \textit{elite pool}} |\{j \in J : i_{S_e}(j) \neq i_S(j)\}| + \sum_{S_d \in \textit{diverse pool}} |\{j \in J : i_{S_d}(j) \neq i_S(j)\}| \right),$$

where n_e (n_d) is the current number of *elite* (*diverse*) solutions and $i_{\texttt{solution}}(j)$ denotes the facility to which customer j is assigned in `solution`.

3.2 Path relinking

As we have already mentioned, the first phase of our heuristic ends with two pools of solutions: the *elite pool* and the *diverse pool*. We recall that in the *elite pool* the solutions are sorted in non-increasing order based on their (estimated) cost.

In the second phase of our heuristic we consider pairs of solutions $(S_{\text{current}}, S_{\text{target}})$ as detailed in Algorithm 1. For each pair a path linking the solutions is explored using our PR procedure. Every time a solution is found whose evaluation is better than the incumbent solution, we update the incumbent. Therefore, the outcome of one execution of the PR is the previous incumbent solution if it was not improved or a new incumbent solution otherwise.

Algorithm 3 `pathRelinking`$(S_{\texttt{current}}, S_{\texttt{target}}, S_{\texttt{incumbent}})$

// initialization
1: nOpenCurrent $\leftarrow$ `countFacilitiesOpen`$(S_{\texttt{current}})$
2: nOpenTarget $\leftarrow$ `countFacilitiesOpen`$(S_{\texttt{target}})$
// end of initialization
3: $\beta \leftarrow$ `RandomSelect`$([0,1])$
4: **while** $(\beta \leq p^{++})$ and (nOpenCurrent $>$ nOpenTarget) **do**
5: $S_{\text{current}} \leftarrow$ `closeFacility`$(S_{\texttt{current}})$
6: nOpenCurrent $\leftarrow$ nOpenCurrent-1
7: $S_{\text{incumbent}} \leftarrow$ `updateIncumbent`$(S_{\texttt{incumbent}}, S_{\texttt{current}})$
8: $\beta \leftarrow$ `RandomSelect`$([0,1])$
9: $S_{\text{current}} \leftarrow$ `exchangeFacilities`$(S_{\texttt{current}})$
10: $S_{\text{incumbent}} \leftarrow$ `updateIncumbent`$(S_{\texttt{incumbent}}, S_{\texttt{current}})$
11: $S_{\text{current}} \leftarrow$ `closingFacilities`$(S_{\texttt{current}})$
12: $S_{\text{incumbent}} \leftarrow$ `updateIncumbent`$(S_{\texttt{incumbent}}, S_{\texttt{current}})$
13: $S_{\text{current}} \leftarrow$ `openingFacilities`$(S_{\texttt{current}})$
14: $S_{\text{incumbent}} \leftarrow$ `updateIncumbent`$(S_{\texttt{incumbent}}, S_{\texttt{current}})$
15: $S_{\text{incumbent}} \leftarrow$ `localSearch_2`$(S_{\texttt{current}}, S_{\texttt{incumbent}})$

Three types of local moves are considered for transforming the current solution into the target: (i) opening a facility that is open in the target solution, (ii) closing a facility that is closed in the target solution, or (iii) interchanging one facility that is open in the current solution but closed in the target by a facility that is closed in the current and open in the target. The above moves include the reassignment of customers if necessary. After the target solution has been reached (which will be the case in line 13 or before), we perform a local search that considers only one type of move: reassignment of customers. We provide all the details below.

The PR procedure is sketched in Algorithm 3. Here after the initialization we perform a `while` loop (lines 3–8). The goal is to start by iteratively closing facilities. Nevertheless, this loop is regulated by a probability p^{++} of moving to other neighborhoods before the current and the target solutions have the same number of open facilities.

We also note that we have considered an alternative version of the PR without lines 3–8. We call *PR1* the procedure depicted in Algorithm 3 and *PR2* the version that omits those lines.

Intuitively, *PR1* should work better than *PR2* since in general fixed costs for opening facilities are large and, therefore, exploring many solutions with a number of opened facilities larger than necessary does not seem very promising. However, our results will show that, on average, PR2 performs slightly better.

Before giving the details about the local moves performed in Algorithm 3 we focus on the incumbent update.

As we will explain in Section 4.2.2 when estimating the cost of a solution we can consider an estimate with higher precision (more time is required to compute it) or an estimate with a lower precision (computationally cheaper). In both versions of the PR, we only estimate the cost of a feasible solution using the higher precision when we wish to update the incumbent. In Algorithm 4 we give the details of this process. In particular, we detail function `updateIncumbent(`$S_{\texttt{incumbent}}$`,` $S_{\texttt{current}}$`)`.

Algorithm 4 `updateIncumbent(`$S_{\texttt{incumbent}}$`,`$S_{\texttt{current}}$`)`

1: $f(S_{\text{current}}) \leftarrow$ estimateCost(S_{current}, lowPrecision)
2: **if** $f(S_{\text{current}}) < (tolerance \times f(S_{\text{incumbent}}))$ **then**
3: $\quad f(S_{\text{current}}) \leftarrow$ estimateCost(S_{current}, highPrecision)
4: $\quad$ **if** $f(S_{\text{current}}) < f(S_{\text{incumbent}})$ **then**
5: $\quad\quad S_{\text{incumbent}} \leftarrow S_{\text{current}}$
6: **return** $S_{\text{incumbent}}$

Note that, since the solution costs are not exactly computed but estimated, a tolerance factor is used to exclude a given solution as a candidate for leading to an incumbent update.

3.2.1 Facility moves

As mentioned above, three different facility moves are considered for reaching the target solution. We next detail these moves.

Closing facilities

When some of the facilities that are open in S_{current} are closed in S_{target} we choose one of them for closing. This is done in a greedy way. First, we estimate the increment in the solution cost by closing a facility. In particular, for some open facility i we compute

$$-f_i + \sum_{j \in J \,|\, i(j)=i} p_j(c_j^{\min} - c_{ij}),$$

where $c_j^{\min}$ represents the minimum service cost for customer j computed among the open facilities in S_{current} (except for facility i that is being checked for closing).

When a facility is eventually closed each customer assigned to it is taken in turn. If the facility to which it is assigned in S_{target} is open, it is directly assigned to that facility. Otherwise, it is assigned greedily using the function defined in the GRASP constructive procedure.

Opening facilities

Facilities that are open in S_{target} but are closed in S_{current} are the candidates for being opened. Again, the facility chosen to be opened is the one producing the least (estimated) increase in the solution cost. Such estimate for a facility i is taken to be

$$f_i + \sum_{j \in J \,|i(j)=i \text{ in } S_{\text{target}}} p_j(c_{ij} - c_{i(j),j}),$$

where $i(j)$ is the facility to which customer j is assigned in S_{current}.

All customers who are assigned to a given plant in S_{target} are automatically assigned to it when it is opened. When doing so infeasibilities with respect to the lower bound on the number of customers assigned to a plant can occur. In this case we use the repair mechanism introduced in Section 3.1.2.

Exchange facilities

The goal in this case is to swap one facility in S_{current} by another one in S_{target}. Suppose that i_1 is open in S_{target} but closed in S_{current}. Additionally, let i_2 be a facility that is closed in S_{target} but open in S_{current}. In this case we can open i_1 and close i_2 in S_{current}. When doing so we reassign to i_1 all the customers initially assigned to i_2. If $\ell_1 > \ell_2$ and there are less than ℓ_1 customers initially assigned to i_2, we end up with a violation in the lower bound constraints. This situation is tackled by considering again the restoration mechanism introduced in Section 3.1.2.

What remains to clarify is the selection of the pair (i_1,i_2) to swap. Let us assume, without loss of generality, that we have a candidate pair (i_1,i_2). We estimate the increment in the solution cost for doing this move as follows. Let J_{i_2} be the set of customers assigned to i_2 in S_{current} (and thus that have to be reassigned due to the closing of i_2). We can partition J_{i_2} into two subsets: $J_{i_2}^1 = \{j \in J_{i_2} \mid i(j) = i_1 \text{ in } S_{\text{target}}\}$ and $J_{i_2}^2 = J_{i_2} \setminus J_{i_2}^1$. The solution cost increment is estimated according to

$$f_{i_1} - f_{i_2} + \sum_{j \in J_{i_2}^1} p_j(c_{i_1 j} - c_{i_2,j}) + \sum_{j \in J_{i_2}^2} p_j(c_j^{\min} - c_{i_2,j}),$$

where $c_j^{\min}$ represents the minimum service cost for customer j computed among the open facilities in $(S_{\text{current}} \setminus \{i_2\}) \cup \{i_1\}$ (excluding facility i_2 that is going to be closed).

3.2.2 Customer moves

After performing all the facility moves (and the corresponding customer reassignments) we perform a local search (Algorithm 3—`LocalSearch_2(`S`)`) involving the customers. We con-

sider a reassignment procedure that is different from the one proposed in the GRASP. The procedure is now based on the following estimation for the variation in cost when reassigning customer j from plant $i_1 = i(j)$ to i_2:

$$c_{i_2 j} - c_{i_1 j} + g_{i_2} \frac{p_j}{z_{i_2} + 1} \max\{z_{i_2} + 1 - K_{i_2}, 0\} - g_{i_1} \frac{p_j}{z_{i_1}} \max\{z_{i_1} - k_{i_1}, 0\}.$$

In the above expression we combine three estimates: (i) the increase of the service costs, (ii) the increase of the expected penalty in plant i_2, which will have one more customer now, and (iii) the reduction of the expected penalty in plant i_1.

4 Computational experience

In this section we report the computational experience carried out for evaluating the new heuristic framework proposed. We start by describing briefly the test instances considered and then we present and discuss several implementation details. Finally, we report the results obtained, firstly for a set of instances of the homogeneous FLPBD (solutions values can be computed exactly and, moreover, optimal values are known) and afterwards for a set of instances of the general problem.

For each instance we focus our attention on three solutions: (i) the solution obtained by the GRASP procedure alone, (ii) the solution obtained when using PR1 after the GRASP, and (iii) the solution obtained when using PR2 after the GRASP.

For evaluating the heuristic in general we compare its results with those obtained with the sampling average approximation approach developed in Albareda-Sambola et al. (2016).

All algorithms were coded in C++ and the computational tests were performed on a machine running an Intel Core i7 4770K with 32GB of RAM.

4.1 Test instances

In our computational experience, we considered two different sets of instances. They are available as supplementary material to this paper. The first one contains instances for the homogeneous case, selected from those generated by Albareda-Sambola et al. (2011). The second set consists of instances generated by Albareda-Sambola et al. (2016) for the general FLPBD. In order to make this paper self-contained, we provide a few details about the instances.

Regarding the instances for the homogeneous FLPBD we recall that they were generated from 11 Traveling Salesman Problem instances available at `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/` namely, `berlin52`, `eil51`, `eil76`, `kroA100`, `kroB100`, `kroC100`, `kroD100`, `kroE100`, `pr76`, `rqt99`, and `st70`. From those instances, small and large FLPBD instances ($|I| = 15/|J| = 30$ and $|I| = 20/|J| = 60$, respectively) were generated. For each combination of plants and customers, the remaining data was generated varying several parameters as follows: (i) three different values for the probability of demand (0.1, 0.5, and 0.9), (ii) three different levels of variability for the set-up costs (0, $\mu/10$, and $\mu/3$, where μ is the expected value set for the setup costs), (iii) low, medium, and high capacity levels, and (iv) two different possibilities for ℓ_i (0 and a value greater than 0). In total, Albareda-Sambola et al. (2011) generated 2754 instances divided into 17 groups (11 groups of small instances and 6 groups of large instances). For the current work, we selected a subset of 306 such instances,

consisting of the 18 instances in each group corresponding to the intermediate value for the variability for the set-up costs ($\mu/3$), low and high capacity levels (the intermediate value was discarded), and $\ell_i > 0, i \in I$.

Concerning the heterogeneous FLPBD, we consider the 306 instances introduced by Albareda-Sambola et al. (2016). These instances were generated from the 306 homogeneous instances just mentioned, considering three types of customers: low-, medium- and high-probability demand customers. The demand probabilities were drawn from $U(0.10, 0.25)$, $U(0.40, 0.60)$, and $U(0.75, 0.90)$ distributions, respectively, for the three customer types. Additionally, three patterns for the demand were considered: in pattern 1 there are 60% of low-probability demand customers, 20% medium and 20% high. In pattern 2 these percentages are 20%, 60% and 20%, respectively; and in pattern 3 the values are 20%, 20% and 60%, respectively. The reader is referred to Albareda-Sambola et al. (2016) for all the details.

Each homogeneous instance is identified by a label of the form `name_id_pr_`γ`_h`, where, `name` is the name of the original TSP instance (with the prefix `L` for large instances), `id` $\in \{1, 2, 3\}$ identifies the instance among the three generated from the same original TSP instance, `pr` $\in \{1, 5, 9\}$ indicates the homogeneous demand probability ('1' corresponds to 0.1, '5' corresponds to 0.5, and '9' corresponds to 0.9), $\gamma \in \{1, 4\}$ is the value of the parameter γ used for generating the capacities of the facilities, and finally, `h` stands for "homogeneous".

Each non-homogeneous instance is identified by a label of the form `name_id_pr_`γ`_n`, with a similar interpretation as above, where now the last suffix `n` stands for non-homogeneous.

4.2 Implementation details

In this section we detail some aspects related to the implementation of the heuristic. In particular, we discuss the exact and simulated computation for the cost of a feasible solution, and we present the parameter tuning.

4.2.1 Evaluating a feasible solution

Recall that the exact evaluation of the recourse function amounts to enumerating, for each open facility i, all the possible sets of demand customers among J_i (or, at least, all those with cardinality greater than K_i). Since $|J_i|$ can be very large, particularly for instances with low demand probabilities, this can become unaffordable, even for one single evaluation.

In the case of facility outsourcing the service costs for a given demand vector are not much involved and thus, even if it takes some considerable CPU time, it is possible to evaluate exactly the cost of a solution by enumerating for each facility all possible subsets of demand customers among the ones assigned to it. This is not the case when customer outsourcing is considered, since the order by which the demand customers call for being served influences the service cost. Therefore, Enumerating all possible scenarios would require to enumerate all possible call sequences for each possible set of demand customers. Accordingly, even if we are willing to allow a large CPU time for evaluating a solution, that might not be possible. Thus, we simulate the expected service cost of an open facility when the number of demand customers assigned *a priori* to that facility is beyond some threshold, $nMax$.

Since the above type of evaluation cannot be used repeatedly, we decided to apply it only once, for the final solution returned by the heuristic.

4.2.2 Estimating the cost of a feasible solution (General FLPBD)

To evaluate a feasible solution it is necessary to evaluate the recourse function, that is, the expected costs of service plus outsourcing. In our case we apply Monte Carlo simulation and use as an estimate of this cost the average cost associated with a sample of scenarios. We do not fix the sample size, instead we keep sampling scenarios until the average converges:

$$100 \times \left| \frac{\text{previous average} - \text{current average}}{\text{previous average}} \right| < \text{precision}, \tag{9}$$

where 'precision' is a small value defined exogenously.

Different strategies exist to accelerate the computation of this average. Preliminary tests suggested the use of antithetic estimators (Breimer et al. 2012). The idea is simple. By taking at each iteration the average of two unbiased estimators of the same value with negative correlation we obtain a new unbiased estimator with smaller variance. In our particular case from each sequence of n pseudo-random numbers U(0,1), $\rho_1, \ldots, \rho_n$, we generate two different scenarios. In the first of them the set of demand customers is defined as $\{j \in J : \rho_j \leqslant p_j\}$ and in the second one it is $\{j \in J : 1 - \rho_j \leqslant p_j\}$. Since the solution costs associated with either scenario are negatively correlated estimators of the solution cost, their mean value is a more efficient estimator than any of them alone. Therefore, our final estimate is the average of such values over a large enough number of simulations.

In the case of facility outsourcing, the order by which the customers call for service is not relevant to determine a scenario and its associated solution cost. However, this is not the case with customer outsourcing. For the latter, in addition to generating the demand customers, we also need to simulate the order by which they call for being served. This is needed in order to know exactly which customers are served from their assigned facilities and which ones are outsourced. As soon as all the above information is available for a particular scenario, it is straightforward to compute the cost of a solution associated with it. Both for facility outsourcing and for customer outsourcing we can repeat this process as many times as we want. Every time we repeat the process we get another antithetic estimate for the solution cost. We stop the process when condition (9) is met.

In our heuristic we consider two different precision levels: high precision and low precision (see Algorithms 1 and 4). In the first phase of our heuristic (Algorithm 1, lines 1–17) we only consider simulating solution costs using high precision. In the second phase (Algorithm 1, lines 18–23) we always consider low precision except when we find a solution that is potentially better than the incumbent. In this case, before updating the incumbent, we deepen the simulation by considering high precision in order to have a more accurate estimate of the potential new incumbent.

Preliminary experiments were carried out using different functions for the approximation of the solution costs such as (i) the exact evaluation of the cost function (see above); (ii) the exact evaluation function associated with the homogeneous case using the average demand probability; (iii) the exact cost evaluation assuming that all customers assigned to the same facility have an equal demand probability given by their average probability; (iv) the cost function simulated as above but always using the lower precision; (v) the cost function simulated as above but always using the higher precision. The obtained results showed that the above described simulation function is the most effective one, among those we tested.

Since for the homogeneous case an exact evaluation of the cost is possible by making use

of the probabilities corresponding to a binomial distribution, when Algorithm 1 is applied to an instance of that problem, function `estimateCost(`S`,`*precision*`)` is replaced by a function that returns the exact value.

4.2.3 Parameter tuning

Before presenting and discussing the results obtained we present the final values chosen for all the parameters involved in our heuristic. Tuning these parameters was quite a relevant step in all this process, which led us through many preliminary tests whose results do not fit the reasonable length of a research paper and are thus omitted. Nevertheless, we briefly mention some of the experiences carried out and then we present the final values chosen for the complete experiments.

- α (parameter defining the range of the RCL in the GRASP). We tested values between 0.1 and 0.4.
- p^{++} (probability ruling the first attempt to closing facilities in PR1). We tried the values 0.1, 0.5, and 0.9.
- *tolerance* (used in Algorithm 4). We tried the values 0.999 and 1.05.

After all the preliminary computations we set the parameters values according to Table 1. We also tested alternative estimations of the cost variations in the moves involved in the algorithms, specifically devised for the customer outsourcing policy. The increase in the computing times did not yield better results and, therefore, these estimates were discarded.

4.3 Results for the homogeneous Case

To assess precisely the quality of the solutions produced by our heuristic, we run a first set of experiments with the homogeneous FLPBD instances. Since in this case optimal solutions are known (see Albareda-Sambola et al. 2011), we can measure the exact percentage optimality gaps of the heuristic solutions obtained.

For each instance 5 runs were performed and the minimum, average, and maximum CPU time (in seconds) and gap (%) were computed.

In Figures 1 and 2 we show aggregated results for the different sets of 18 instances associated with each original TSP instance. For each set we present three vertical bars: the first one gives the average of the 5-run average gaps (%) with respect to the optimal solutions obtained in Albareda-Sambola et al. (2011) after executing the GRASP procedure alone. The following two bars depict a similar average when each of the two variants of the PR are added to the process. The values used for drawing these figures are given in Tables 2 and 3 of the Appendix, which present average results over the set of homogeneous instances associated with each original TSP instance for the facility outsourcing and the customer outsourcing policies, respectively. In particular, for each tested heuristic (GRASP, GRASP+PR1 and GRASP+PR2) $\%Gap$ indicates the average percentage gap at termination, relative to the optimal value, $\%Opt$ the average percentage of optimal solutions found over all the runs, and *CPU(secs.)* the average computing time in seconds.

As we can conclude from Figures 1 and 2, and Tables 2 and 3, the best feasible solution obtained with GRASP is on average already good. In fact, the largest average gaps obtained with GRASP were 1.81% for facility outsourcing and 1.62% for customer outsourcing. When

Table 1: Values set for the parameters.

Parameter	Usage	Description	value
`nElite`	GRASP (Algorithm 1)	Maximum number of solutions in the *elite_pool*	15
`nDiverse`	GRASP (Algorithm 1)	Maximum number of solutions in the *diverse_pool*	15
`max_iter`	GRASP (Algorithm 1)	Number of GRASP iterations	500
α	GRASP (Algorithm 2)	Range for the RCL	0.15
p^{+}	GRASP (Algorithm 2)	Probability ruling the process of opening further facilities while the stopping criteria are not met	0.9
`nMoves`	GRASP (`LocalSearch_1(`S`)`)	Maximum number of times the same reassignment can be done in the interchange and reassignment moves	2
`rep`	Algorithm 1	Number of times each elite solution will be used	2
p^{++}	PR1	Probability of keeping closing facilities in the first phase of converting the current solution into the target	0.9
`lower precision`	PR 1 and PR2 (Algorithm 4)	Lower precision for simulating the cost of an *a priori* solution	5E-4
`higher precision`	GRASP, PR1 and PR2 (Algorithms 1 and 4)	Higher precision for simulating the cost of an *a priori* solution	1E-5
`tolerance`	GRASP, PR1 and PR2 (Algorithms 1 and 4)	Tolerance used to exclude a given solution as a candidate to become an incumbent	1.05
`iterMin`	Antithetic estimator	Minimum number of random sequences to generate independently of the stopping criteria	500
`ntop`	Solution evaluation (customer outsourcing)	Maximum number of demand customers assigned *a priori* to one facility for which we compute the service cost exactly	20

the proposed PR procedures are applied after GRASP we observe a clear decrease in the average gaps. Additionally, for the homogeneous case the two variants of the PR seem to produce very similar average results although, the detailed results observed in Tables 2 and 3 show that *PR1* is able to find the optimal solution in more instances than *PR2*.

In Figures 3 and 4 we can observe the average CPU time required by each of the three procedures considered. Accordingly, when we consider the GRASP followed by one of the PR variants we should add both averages to obtain the average CPU time required by GRASP

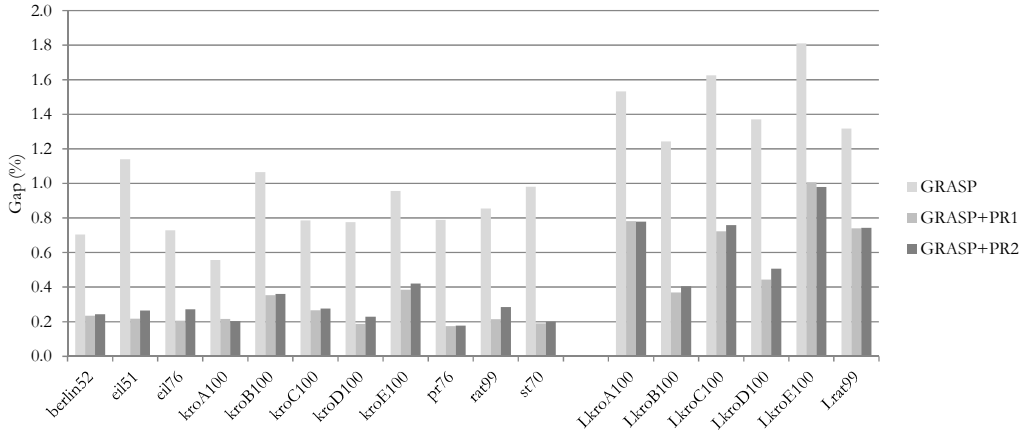

Figure 1: Homogeneous case — Facility outsourcing — gap (%).

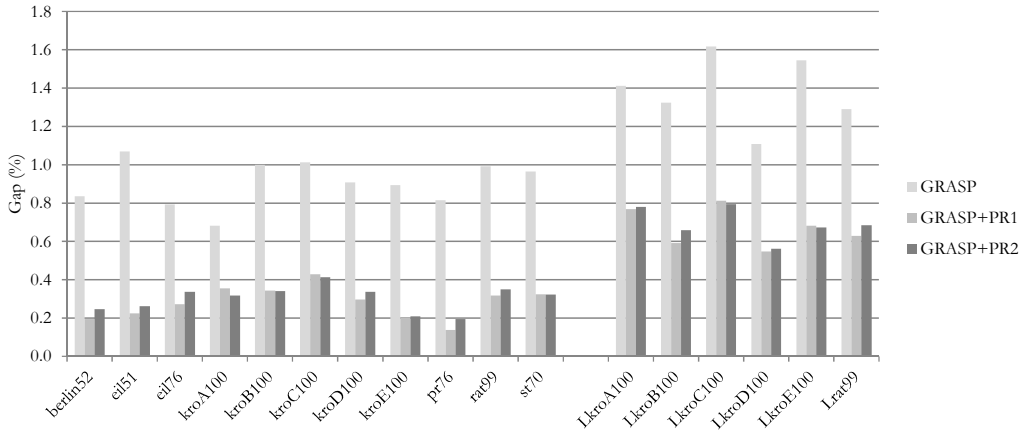

Figure 2: Homogeneous case — Customer outsourcing — gap (%).

together with PR.

Observing the average CPU times depicted in Figures 3 and 4 we conclude that they are negligible, even for the larger instances since they never exceed 0.7 seconds on average for the two considered outsourcing policies.

Summing up the information depicted in Figures 1–4 we can say that the new heuristic framework proposed is extremely effective for the homogeneous FLPBD. Although the gaps obtained using GRASP are already small, both variants of the PR are able to significantly reduce them further. Moreover, the CPU time required by GRASP increases significantly with the size of the instances, which is not the case with PR. Indeed, the number of iterations and the size of the pools of solutions do not change with the size of the instance.

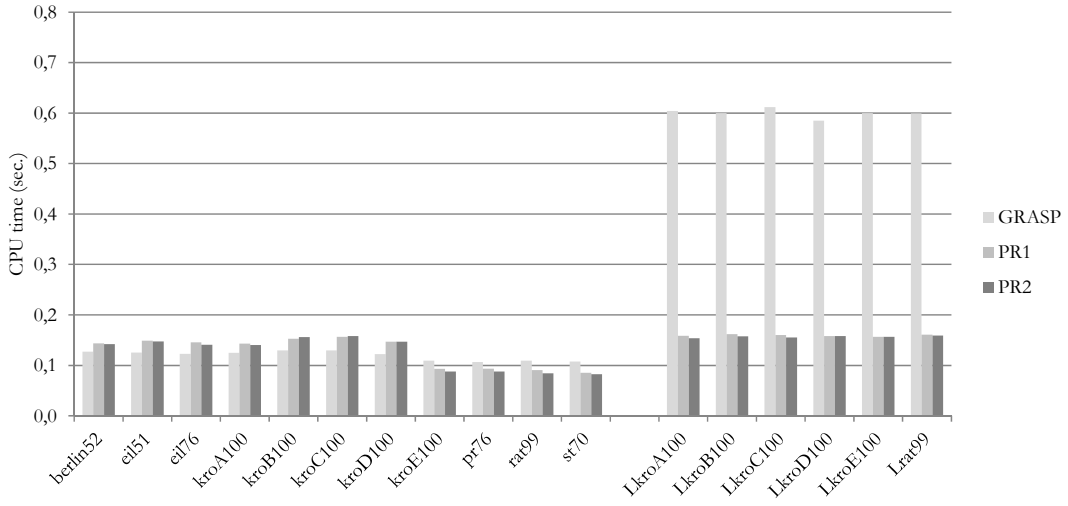

Figure 3: Homogeneous case — Facility outsourcing — CPU (sec.).

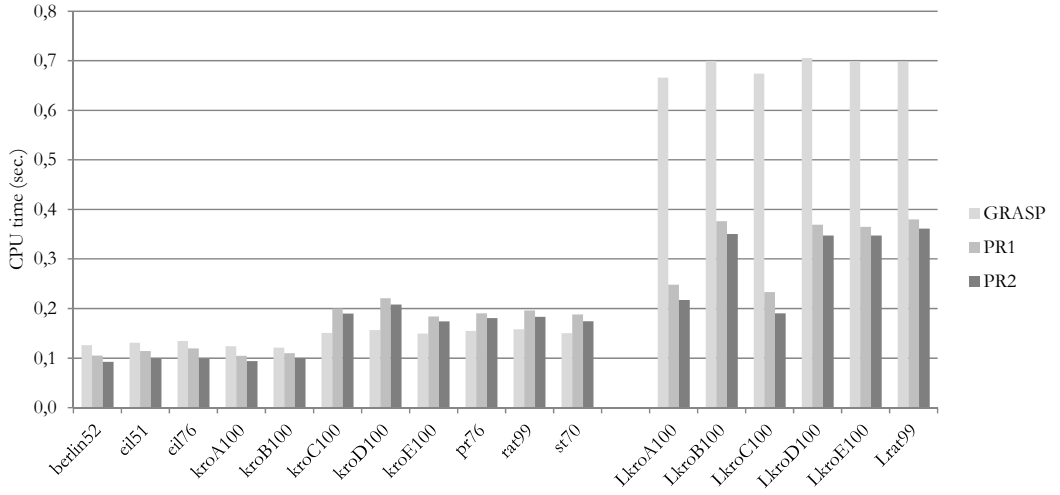

Figure 4: Homogeneous case — Customer outsourcing — CPU (sec.).

4.4 Results for the general problem

For the problem with non-homogeneous demand probabilities we compare the results with those obtained by the sample average approximation (SAA) procedure proposed in Albareda-Sambola et al. (2016) since an enumerative algorithm is not computationally affordable.

A summary of the results can be observed in Figures 5–8. The data depicted in these figures can be found in the Appendix, Tables 4 and 5. These tables give average values of $\%Gap$ and *CPU(secs)* for the instances of the general non-homogeneous case. Since optimal solutions are not known for these instances the gaps are computed with respect to the best solution found for each instance over all runs of all the tested algorithms. The extensive results (with the specific instances tested for customer outsourcing) can be found in the Appendix—Tables 6–8.

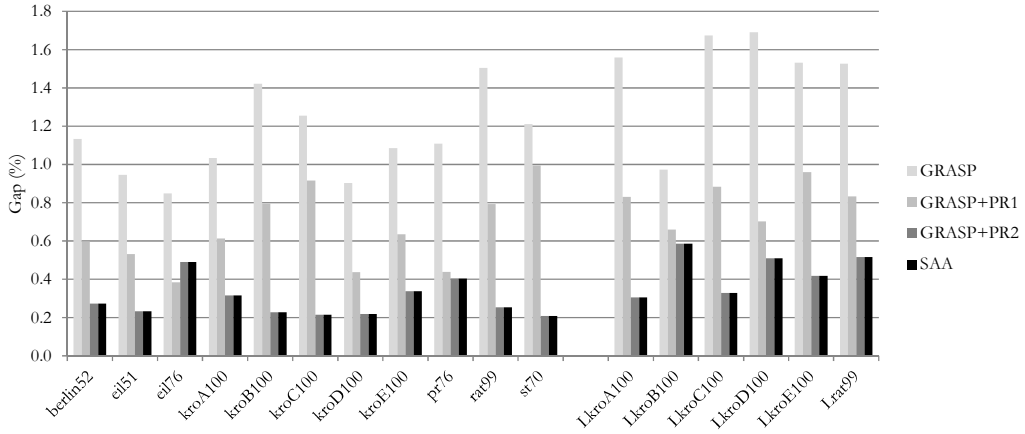

Figure 5: Non-homogeneous case — Facility outsourcing — gap (%).

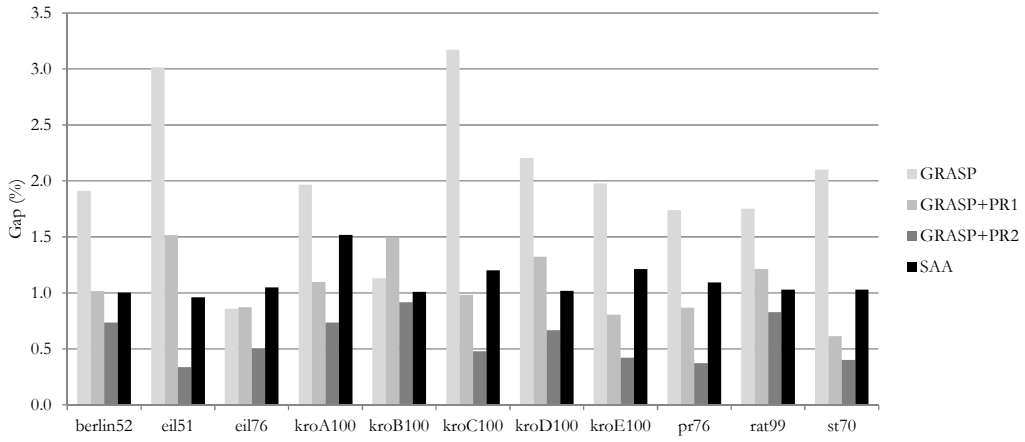

Figure 6: Non-homogeneous case — Customer outsourcing — gap (%).

For each instance tested we obtained four values, namely those associated with the best feasible solution obtained using: (i) the SAA procedure, (ii) GRASP alone, (iii) GRASP with PR1, and (iv) GRASP with PR2. We take the smallest among these four values as the best-known upper bound for the instance. Then, we compute the gaps of the other three values with respect to this one.

The average such gaps computed for the set of instances associated with each original TSP instance can be observed in Figures 5 and 6. Looking into these figures we can draw two interesting conclusions: (1) the gaps relative to the best-known solutions are rather small. The largest average gap for facility outsourcing was 1,69% and for customer outsourcing 3,17%; (2) Using SAA the average gap is above zero which is an indication that not always the best feasible solution was obtained with SAA but with GRASP and PR.

The above observations have even more impact when we look into Figures 7 and 8. In these figures we can observe the average CPU times (in seconds) required by each of the four tested procedures. When we consider the GRASP followed by one of the PR variants,

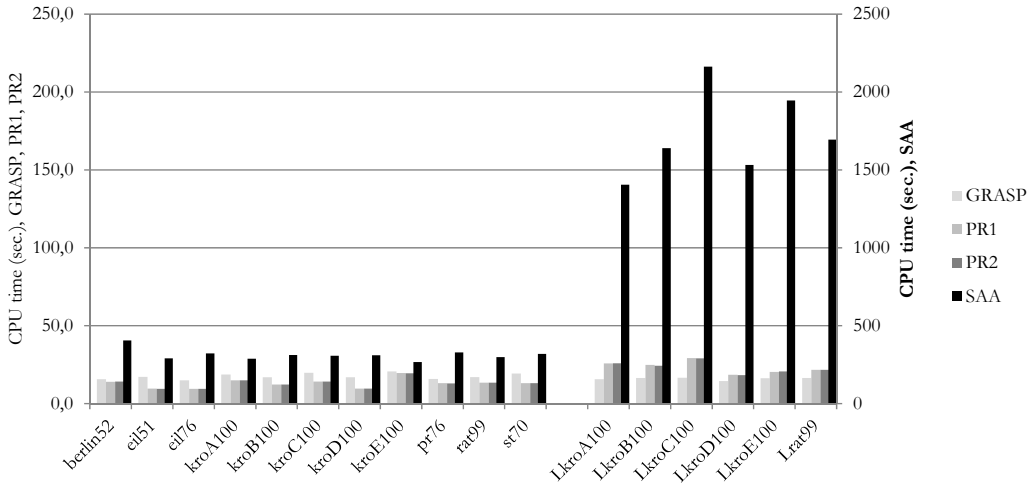

Figure 7: Non-homogeneous case — Facility outsourcing — CPU (sec.).

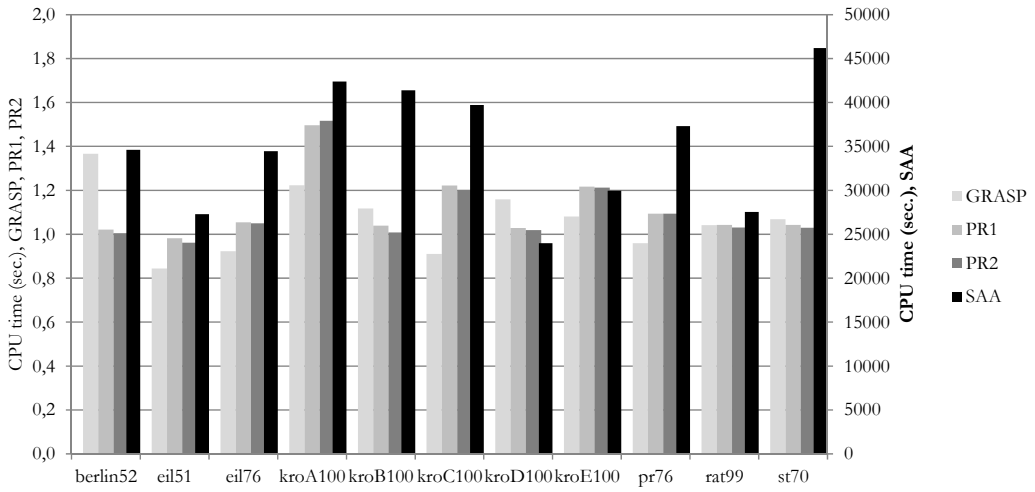

Figure 8: Non-homogeneous case — Customer outsourcing — CPU (sec.).

we should add both averages to obtain the average CPU times required by GRASP together with PR. In the case of SAA, since the magnitude of the CPU times was totally different we considered a secondary axis (the right-hand-side axis) for indicating those values.

Observing the average CPU times depicted in Figures 7 and 8 we conclude that, like for the homogeneous case, they are negligible when using GRASP combined with PR. However, they are quite significant when using SAA.

All the computations performed indicate that the heuristic framework proposed is robust as evidenced by the outcome of the five runs executed for each instance, which yielded similar results.

We can also observe a slight superiority of *PR2* on average, even if *PR1* is able to find the optimal solution in more instances.

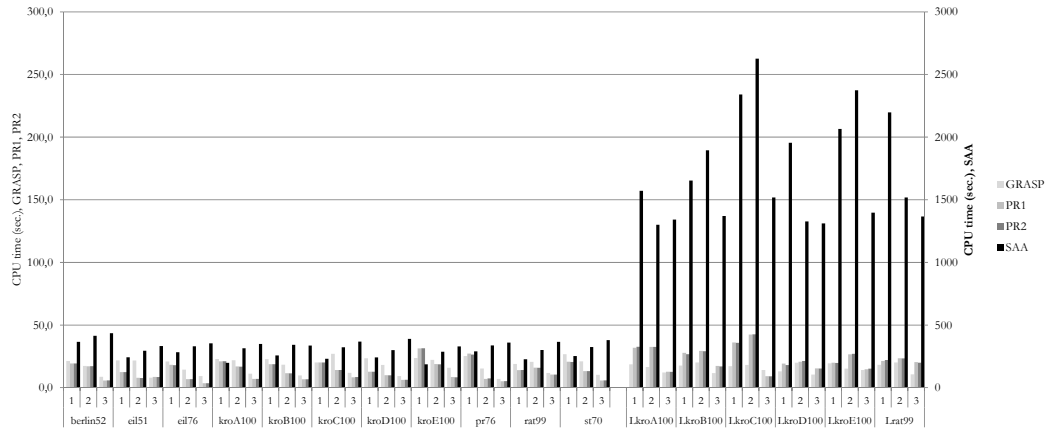

Figure 9: Non-homogeneous case — Facility outsourcing — CPU (sec.).

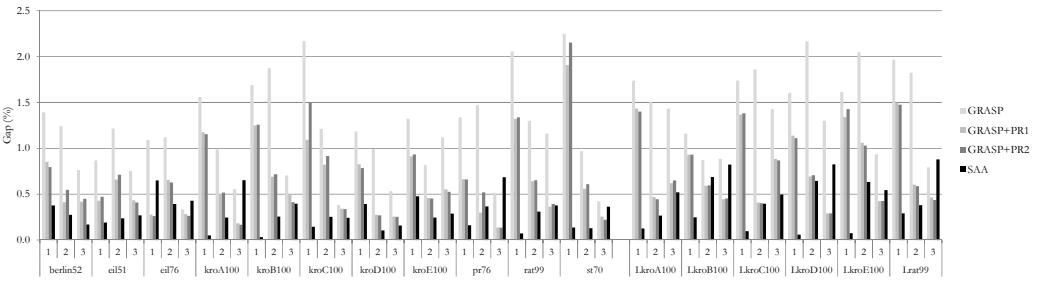

Figure 10: Non-homogeneous case — Facility outsourcing — gap (%).

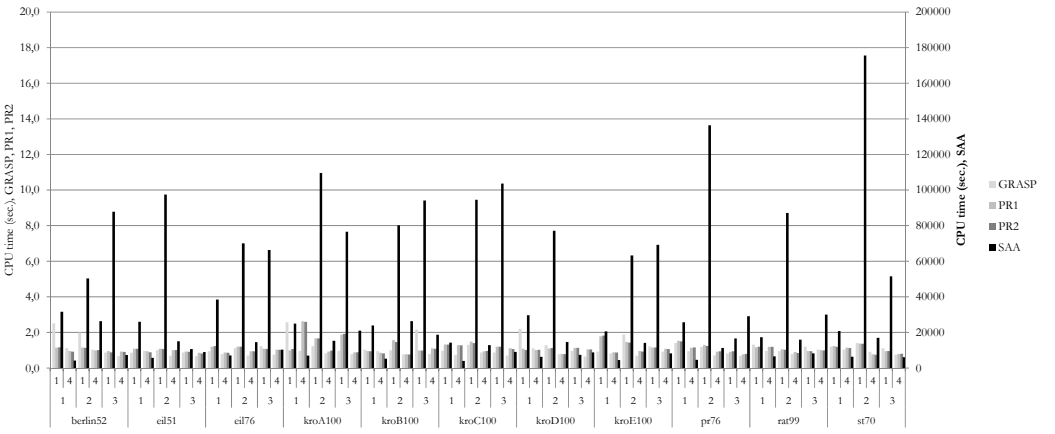

Figure 11: Non-homogeneous case — Customer outsourcing — CPU (sec.).

5 Conclusions

In this paper we have presented a heuristic that combines GRASP with path relinking for the general case of the Facility Location Problem with Bernoulli Demands with two different

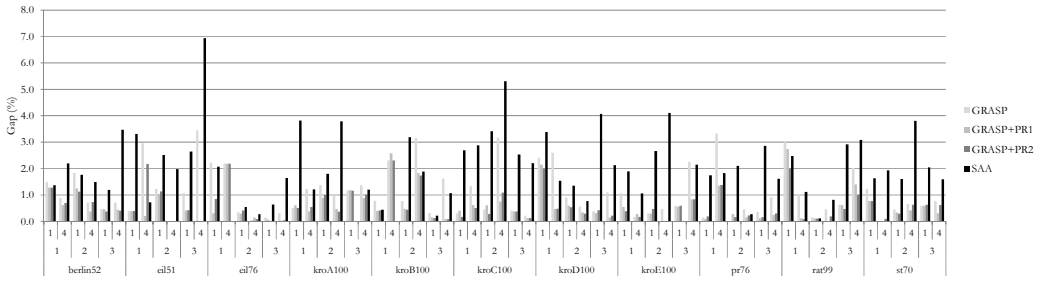

Figure 12: Non-homogeneous case — Customer outsourcing — gap (%).

outsourcing policies: facility outsourcing and customer outsourcing. In both cases we have applied two variants of this heuristic both to the homogeneous and to the general case of the problem. Since the problem with homogeneous demand probabilities can be exactly solved quite efficiently, homogeneous instances were used to give a precise evaluation of the quality of the solutions provided by the algorithm. In the general case the solutions were compared with those provided by a sample average approximation procedure. In both cases the quality of the solutions is high and the algorithm found them in short CPU times, specially if they are compared with those required by sample average approximation. To the best of our knowledge this is the first time that an algorithm is proposed in the literature to solve the general case of the Facility Location Problem with Bernoulli Demands.

Acknowledgements

This research has been partially supported by the *Spanish Ministry of economy and competitiveness*, Project MTM2015-63779-R and by the Portuguese Science Foundation (FCT—Fundação para a Ciência e Tecnologia) under the project UID/MAT/04561/2013 (CMAF-CIO/FCUL). The authors thank the two anonymous reviewers for the valuable comments, which helped improving the manuscript.

References

M. Albareda-Sambola, E. Fernández, and F. Saldanha-da-Gama. The facility location problem with Bernoulli demands. *Omega*, 39:335–345, 2011.

M. Albareda-Sambola, E. Fernández, and F. Saldanha-da-Gama. Outsourcing policies for the facility location problem with uncertain demand. *In preparation*, 2016.

F. Ballestín and R. Leus. Resource-constrained project scheduling for timely project completion with stochastic activity durations. *Production and Operations Management*, 18: 459–474, 2009.

M. Bieniek. A note on the facility location problem with stochastic demands. *Omega*, 55: 53–60, 2015.

M.B. Freimer, J.T. Linderoth, and D.J. Thomas. The impact of sampling methods on bias and variance in stochastic linear programs. *Computational Optimization and Applications*, 51:51–75, 2012.

D. Cattaruzza, N. Absi, D. Feillet, and T. Vidal. A memetic algorithm for the Multi Trip Vehicle Routing Problem. *European Journal of Operational Research*, 236(3):833–848, 2014.

I. Correia and F. Saldanha-da-Gama. Facility location under uncertainty. In G. Laporte, S. Nickel, and F. Saldanha-da-Gama, editors, *Location Science*, pages 177–203. Springer, Berlin-Heidelberg, 2015.

H. Delmaire, J. A. Díaz, E. Fernández, and M. Ortega. Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem. *INFOR*, 37: 194–225, 1999.

J. A. Díaz and E. Fernández. Hybrid scatter search and path relinking for the capacitated p-median problem. *European Journal of Operational Research*, 169:570–585, 2006.

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

P. Festa and M. G. C. Resende. Hybridizations of GRASP with path-relinking. *Studies in Computational Intelligence*, 434:135–155, 2013.

F. Glover. Tabu search and adaptive memory programing - Advances, applications and challenges. In R.S. Barr, R.V. Helgasson, and J.L. Kennington, editors, *Interfaces in Computer Science and Operations Research*, pages 1–75. Springer, 1997.

F. Glover and M. Laguna. *Tabu Search*. Kluwer, Norwell, Massachusetts, 1997.

F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29:653–684, 2000.

H. Held and D. L. Woodruff. Heuristics for multi-stage interdiction of stochastic networks. *Journal of Heuristics*, 11:483–500, 2005.

M. Laguna and R. Martí. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52, 1999.

F. V. Louveaux. Stochastic location analysis. *Location Science*, 1:127–154, 1993.

M. T. Melo, S. Nickel, and F. Saldanha-da-Gama. Dynamic multi-commodity capacitated facility location: a mathematical modeling framework for strategic supply chain planning. *Computers & Operations Research*, 33:181–208, 2006.

M. T. Melo, S. Nickel, and F. Saldanha-da-Gama. Facility location and supply chain management — a review. *European Journal of Operational Research*, 196:401–412, 2009.

M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures: Advances and applications. In J.-Y. Potvin and M. Gendreau, editors, *Handbook of Metaheuristics*, 2nd Edition. Springer, New York, 2008.

M. G. C. Resende and C. C. Ribeiro. GRASP: Greedy randomized adaptive search procedures. In E. K. Burke and G. Kendall, editors, *Search Methodologies*, pages 287–312. Springer, New York, 2013.

M. G. C. Resende and R. F. Werneck. A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, 174:54–68, 2006.

R. Z. Ríos-Mercado and E. Fernández, A reactive GRASP for a commercial territory design problem with multiple balancing requirements. *Computers & Operations Research*, 36(3): 755–776, 2009.

L. V. Snyder. Facility location under uncertainty: a review. *IIE Transactions*, 38:547–564, 2006.

T. Vidal, T. G. Crainic, M .Gendreau, and C. Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489, 2013.

S. Wang, J. Watada, and W. Pedrycz. Value-at-risk-based two-stage fuzzy facility location problems. *IEEE Transactions on Industrial Informatics*, 5:465–482, 2009.

Appendix

In this appendix we present some detailed results.

In Tables 2 and 3, for each set of homogeneous FLPBD instances associated with one original TSP instance we present the average gap (%*Gap*) with respect to the optimal solutions obtained in Albareda-Sambola et al. (2011), the average percentage of optimal solutions found (%*Opt*) over all the runs on each instance, and the CPU time(*CPU(secs)*). In Tables 4 and 5 we can find the values of %*Gap* and *CPU(secs)* for the instances of the general non-homogeneous case. Since optimal solutions are not known for these instances the gaps are now computed with respect to the best solution found for each instance over all runs of all the tested algorithms.

In Tables 6–8 we detail the results for the general case. In the first two tables we have results for facility outsourcing (small and large instances in different tables). SAA was unable to solve the large instances with customer outsourcing. Therefore, in this case we only present results for small instances. Since we performed five runs of the GRASP with PR for every instance, each line in these three tables corresponds to the averages of five-run averages. For all instances SAA was executed only once.

Table 2: Homogeneous case — facility outsourcing — average results.

	%Gap (% Opt)			CPU (sec.)		
	GRASP	GRASP+PR1	GRASP+PR2	GRASP	GRASP+PR1	GRASP+PR2
berlin52	0.13 (27.8)	0.14 (44.4)	0.14 (41.1)	0.71	0.23	0.24
eil51	0.13 (0.0)	0.15 (30.0)	0.15 (28.9)	1.14	0.22	0.27
eil76	0.12 (16.7)	0.15 (46.7)	0.14 (36.7)	0.73	0.21	0.27
kroA100	0.12 (20.0)	0.14 (46.7)	0.14 (45.6)	0.56	0.22	0.20
kroB100	0.13 (16.7)	0.15 (47.8)	0.16 (44.4)	1.07	0.35	0.36
kroC100	0.13 (16.7)	0.16 (40.0)	0.16 (37.8)	0.79	0.27	0.28
kroD100	0.12 (20.0)	0.15 (45.6)	0.15 (42.2)	0.78	0.19	0.23
kroE100	0.11 (10.0)	0.09 (47.8)	0.09 (48.9)	0.96	0.39	0.42
pr76	0.11 (16.7)	0.09 (35.6)	0.09 (30.0)	0.79	0.17	0.18
rat99	0.11 (2.2)	0.09 (46.7)	0.08 (45.6)	0.85	0.21	0.28
st70	0.11 (5.6)	0.09 (38.9)	0.08 (31.1)	0.98	0.19	0.20
LkroA100	0.60 (11.1)	0.16 (26.7)	0.15 (28.9)	1.53	0.78	0.78
LkroB100	0.60 (0.0)	0.16 (27.8)	0.16 (25.6)	1.24	0.37	0.41
LkroC100	0.61 (21.1)	0.16 (25.6)	0.16 (25.6)	1.63	0.72	0.76
LkroD100	0.58 (0.0)	0.16 (30.0)	0.16 (30.0)	1.37	0.44	0.51
LkroE100	0.60 (4.4)	0.16 (14.4)	0.16 (15.6)	1.81	1.01	0.98
Lrat99	0.60 (5.6)	0.16 (16.7)	0.16 (15.6)	1.32	0.74	0.74

Table 3: Homogeneous case — customer outsourcing — average gaps and computing times

	%Gap (% Opt)			CPU (sec.)		
	GRASP	GRASP+PR1	GRASP+PR2	GRASP	GRASP+PR1	GRASP+PR2
berlin52	0.84 (7.8)	0.20 (35.6)	0.25 (30.0)	0.13	0.11	0.09
eil51	1.07 (5.6)	0.22 (35.6)	0.26 (36.7)	0.13	0.11	0.10
eil76	0.79 (15.6)	0.27 (40.0)	0.34 (44.4)	0.13	0.12	0.10
kroA100	0.68 (15.6)	0.35 (47.8)	0.32 (45.6)	0.12	0.10	0.09
kroB100	1.00 (11.1)	0.34 (34.4)	0.34 (36.7)	0.12	0.11	0.10
kroC100	1.01 (16.7)	0.43 (41.1)	0.41 (40.0)	0.15	0.20	0.19
kroD100	0.91 (11.1)	0.30 (38.9)	0.34 (42.2)	0.16	0.22	0.21
kroE100	0.89 (12.2)	0.20 (40.0)	0.21 (40.0)	0.15	0.18	0.17
pr76	0.82 (16.7)	0.14 (37.8)	0.19 (26.7)	0.15	0.19	0.18
rat99	0.99 (11.1)	0.32 (40.0)	0.35 (35.6)	0.16	0.20	0.18
st70	0.96 (0.0)	0.32 (13.3)	0.32 (7.8)	0.15	0.19	0.17
LkroA100	1.41 (6.7)	0.77 (16.7)	0.78 (16.7)	0.67	0.25	0.22
LkroB100	1.32 (0.0)	0.59 (16.7)	0.66 (16.7)	0.70	0.38	0.35
LkroC100	1.62 (11.1)	0.81 (14.4)	0.79 (15.6)	0.67	0.23	0.19
LkroD100	1.11 (0.0)	0.55 (16.7)	0.56 (13.3)	0.71	0.37	0.35
LkroE100	1.54 (0.0)	0.68 (13.3)	0.67 (13.3)	0.70	0.37	0.35
Lrat99	1.29 (11.1)	0.63 (16.7)	0.68 (15.6)	0.70	0.38	0.36

Table 4: Non-homogeneous case — facility outsourcing — average gaps and computing times

	Gap (%)				CPU (sec.)			
	GRASP	GRASP+PR1	GRASP+PR2	SAA	GRASP	GRASP+PR1	GRASP+PR2	SAA
berlin52	1.13	0.56	0.60	0.27	15.74	14.03	14.13	404.81
eil51	0.95	0.51	0.53	0.23	17.18	9.64	9.55	290.49
eil76	0.85	0.40	0.38	0.49	14.86	9.56	9.44	322.52
kroA100	1.03	0.62	0.61	0.32	18.65	14.99	14.96	287.21
kroB100	1.42	0.81	0.80	0.23	16.93	12.26	12.26	311.72
kroC100	1.25	0.75	0.92	0.21	19.68	14.22	14.24	306.77
kroD100	0.90	0.45	0.44	0.22	16.94	9.63	9.60	310.06
kroE100	1.09	0.64	0.64	0.34	20.61	19.58	19.52	267.27
pr76	1.11	0.37	0.44	0.40	15.87	13.14	13.02	329.08
rat99	1.50	0.78	0.79	0.25	17.08	13.49	13.48	297.70
st70	1.21	0.91	0.99	0.21	19.30	13.20	13.17	319.16
LkroA100	1.56	0.84	0.83	0.30	15.75	25.73	25.91	1404.45
LkroB100	0.97	0.65	0.66	0.59	16.50	24.82	24.31	1639.59
LkroC100	1.67	0.89	0.88	0.33	16.52	29.24	29.11	2161.80
LkroD100	1.69	0.71	0.70	0.51	14.45	18.47	18.19	1531.23
LkroE100	1.53	0.94	0.96	0.42	16.26	20.42	20.65	1945.34
Lrat99	1.53	0.85	0.83	0.52	16.39	21.72	21.77	1693.76

Table 5: Non-homogeneous case — facility outsourcing —average gaps and computing times

	Gap (%)				CPU (sec.)			
	GRASP	GRASP+PR1	GRASP+PR2	SAA	GRASP	GRASP+PR1	GRASP+PR2	SAA
berlin52	1.91	1.02	0.73	1.00	1.37	1.02	1.00	34615.26
eil51	3.02	1.52	0.34	0.96	0.84	0.98	0.96	27304.23
eil76	0.86	0.87	0.50	1.05	0.92	1.05	1.05	34450.98
kroA100	1.97	1.10	0.73	1.52	1.22	1.50	1.52	42384.18
kroB100	1.13	1.49	0.92	1.01	1.12	1.04	1.01	41402.15
kroC100	3.17	0.98	0.48	1.20	0.91	1.22	1.20	39698.78
kroD100	2.21	1.32	0.67	1.02	1.16	1.03	1.02	23983.36
kroE100	1.98	0.81	0.42	1.21	1.08	1.22	1.21	29975.07
pr76	1.74	0.87	0.37	1.09	0.96	1.09	1.09	37301.19
rat99	1.75	1.21	0.83	1.03	1.04	1.04	1.03	27534.38
st70	2.10	0.61	0.40	1.03	1.07	1.04	1.03	46208.92

Table 6: Non-homogeneous case — Facility outsourcing — small instances.

	Percent gaps							Computing times						
	GRASP		GRASP+PR1		GRASP+PR2		SAA	GRASP		GRASP+PR1		GRASP+PR2		SAA
	avg	max	avg	max	avg	max	avg	avg	max	avg	max	avg	max	avg
berlin52_1_1_1_n	0.80	1.51	0.96	1.85	0.80	1.51	0.00	24.17	26.78	5.59	9.04	4.73	6.66	551.77
berlin52_1_1_4_n	1.30	1.96	0.71	1.57	0.85	1.57	1.61	24.02	26.06	15.07	20.61	16.13	22.36	207.83
berlin52_1_5_1_n	1.39	2.24	1.01	1.55	1.00	1.96	0.00	11.87	12.56	2.46	3.26	2.42	2.80	566.48
berlin52_1_5_4_n	1.90	3.19	0.35	0.73	1.02	2.05	0.13	13.12	14.90	7.29	10.88	8.17	9.38	245.00
berlin52_1_9_1_n	0.09	0.13	0.13	0.32	0.08	0.13	0.20	4.76	5.13	1.21	1.44	1.18	1.28	609.86
berlin52_1_9_4_n	0.67	1.08	0.11	0.57	0.11	0.57	0.52	7.68	8.19	12.44	15.49	12.99	16.65	301.45
berlin52_2_1_1_n	1.70	2.24	1.77	2.12	1.67	2.12	0.00	12.77	14.06	4.99	5.50	4.83	5.61	440.20
berlin52_2_1_4_n	2.65	3.04	1.16	1.48	1.00	1.48	0.00	18.34	19.87	8.53	10.42	7.65	9.46	193.83
berlin52_2_5_1_n	0.60	0.90	0.29	0.53	0.25	0.42	0.00	28.73	31.81	6.53	7.09	6.39	7.07	554.27
berlin52_2_5_4_n	1.48	2.30	0.00	0.00	0.00	0.00	1.45	16.31	17.38	31.24	39.34	31.14	39.14	211.45
berlin52_2_9_1_n	0.12	0.23	0.09	0.12	0.07	0.12	0.00	14.47	15.15	0.97	1.39	0.98	1.28	531.52
berlin52_2_9_4_n	1.84	1.84	0.38	0.98	0.78	1.09	0.00	9.08	9.88	8.70	10.57	8.94	12.15	255.32
berlin52_3_1_1_n	1.14	1.79	0.50	1.08	0.44	1.06	0.57	18.03	19.03	6.71	8.58	6.71	8.66	595.57
berlin52_3_1_4_n	0.77	1.21	0.01	0.04	0.01	0.03	0.07	30.42	31.65	74.96	89.72	76.20	93.10	204.58
berlin52_3_5_1_n	0.89	1.22	0.71	1.10	0.89	1.24	0.02	11.52	12.25	4.06	5.10	4.12	5.24	719.02
berlin52_3_5_4_n	1.19	1.61	0.12	0.61	0.12	0.61	0.04	22.14	23.70	50.99	57.24	51.05	62.93	191.26
berlin52_3_9_1_n	0.53	0.99	0.49	0.82	0.34	0.82	0.29	4.71	5.29	1.10	1.17	1.06	1.10	684.87
berlin52_3_9_4_n	1.33	2.91	1.31	2.87	1.31	2.87	0.00	11.23	12.58	9.77	12.13	9.69	12.17	222.37
eil51_1_1_1_n	0.84	0.93	0.44	0.49	0.55	0.93	0.00	12.52	14.59	2.53	2.94	2.30	2.84	381.18
eil51_1_1_4_n	2.08	2.08	0.70	0.80	0.80	0.80	0.00	19.93	22.20	18.06	20.72	16.90	22.53	184.89
eil51_1_5_1_n	1.30	1.46	1.30	1.46	1.56	2.16	0.00	34.38	35.66	1.35	1.51	1.46	1.66	428.17
eil51_1_5_4_n	0.33	0.33	0.20	0.33	0.40	0.70	0.27	14.77	15.70	8.01	9.13	8.06	9.69	194.38
eil51_1_9_1_n	0.61	0.82	0.70	0.93	0.49	0.93	0.72	6.40	6.75	0.82	0.97	0.83	0.98	444.36
eil51_1_9_4_n	1.55	3.99	1.06	3.33	0.97	2.89	0.00	6.91	7.47	7.27	8.84	6.69	8.30	262.24
eil51_2_1_1_n	0.09	0.23	0.02	0.11	0.07	0.23	0.31	23.06	24.11	6.02	7.46	4.51	6.08	344.53
eil51_2_1_4_n	0.73	0.73	0.00	0.00	0.00	0.00	0.83	25.67	29.36	25.61	29.63	29.18	32.25	116.56
eil51_2_5_1_n	1.88	2.34	1.43	2.19	1.33	2.34	0.00	29.44	30.53	4.29	4.88	4.11	4.58	449.50
eil51_2_5_4_n	1.67	1.67	0.23	1.16	0.20	1.02	0.00	15.59	15.97	24.88	29.45	24.45	26.20	152.95
eil51_2_9_1_n	0.54	0.58	0.33	0.58	0.39	0.58	0.36	6.57	6.94	0.77	0.84	0.77	0.84	440.48
eil51_2_9_4_n	1.13	3.01	0.01	0.01	0.01	0.01	0.00	12.27	13.20	23.91	28.70	23.96	29.37	176.32
eil51_3_1_1_n	1.07	2.60	1.06	2.60	1.06	2.60	0.00	27.03	29.30	5.58	9.81	4.96	6.67	281.70
eil51_3_1_4_n	0.39	0.39	0.35	0.39	0.35	0.39	0.00	22.64	24.72	17.33	20.83	16.88	21.55	151.58
eil51_3_5_1_n	1.00	1.36	0.40	0.62	0.43	0.62	0.94	22.31	23.36	3.02	3.49	2.98	3.33	395.92
eil51_3_5_4_n	1.13	1.16	0.40	0.62	0.33	0.62	0.21	13.83	15.16	5.36	6.49	5.55	6.69	147.55
eil51_3_9_1_n	0.51	0.79	0.33	0.70	0.43	0.79	0.21	8.15	8.38	1.41	1.84	1.41	1.91	493.80
eil51_3_9_4_n	0.17	0.86	0.17	0.86	0.17	0.86	0.33	7.67	8.19	17.22	22.29	16.89	22.11	182.78
eil76_1_1_1_n	0.74	0.74	0.48	0.74	0.45	0.74	0.13	16.31	16.86	5.88	7.24	4.90	6.81	439.73
eil76_1_1_4_n	0.78	0.78	0.00	0.00	0.00	0.00	1.10	17.53	19.27	17.62	20.68	18.50	22.02	173.95
eil76_1_5_1_n	0.61	1.00	0.57	1.00	0.51	1.00	0.00	7.77	8.97	1.37	1.63	1.28	1.56	492.25
eil76_1_5_4_n	0.64	0.64	0.92	1.04	0.97	1.28	0.00	11.22	12.23	12.86	15.50	13.27	15.77	153.09
eil76_1_9_1_n	0.76	1.17	0.77	1.35	0.68	1.35	0.18	6.84	7.11	1.21	1.34	1.24	1.34	508.63
eil76_1_9_4_n	0.05	0.05	0.02	0.05	0.00	0.00	0.71	11.27	12.25	2.52	3.22	2.55	3.40	208.41
eil76_2_1_1_n	1.10	1.92	0.33	0.89	0.26	0.89	0.09	18.74	19.95	4.66	5.15	4.70	5.61	351.85
eil76_2_1_4_n	2.84	3.34	0.14	0.70	0.14	0.70	0.48	24.41	26.45	44.66	59.81	44.88	59.39	131.19
eil76_2_5_1_n	2.03	3.21	0.92	1.79	0.97	1.79	0.21	30.79	31.43	2.15	2.84	2.12	2.76	404.98
eil76_2_5_4_n	1.66	3.53	0.81	1.38	0.73	1.38	0.00	12.74	13.79	13.75	20.71	13.60	20.24	182.61
eil76_2_9_1_n	0.83	1.60	0.63	1.22	0.55	0.93	0.00	12.83	13.52	1.29	1.47	1.28	1.38	452.79
eil76_2_9_4_n	0.07	0.07	0.00	0.02	0.00	0.00	0.24	11.99	12.55	5.24	6.25	5.34	5.78	196.35
eil76_3_1_1_n	0.75	1.86	0.65	1.86	0.61	1.86	0.60	24.61	24.94	6.03	8.66	5.65	7.69	414.45
eil76_3_1_4_n	0.32	0.46	0.06	0.29	0.12	0.29	1.50	23.63	25.61	30.76	35.13	28.54	34.45	187.77
eil76_3_5_1_n	0.79	1.85	0.28	1.08	0.17	0.77	0.48	15.83	16.73	1.55	1.89	1.54	1.76	511.25
eil76_3_5_4_n	1.01	2.25	0.44	0.62	0.42	0.93	1.66	8.03	8.54	9.24	12.61	9.26	12.60	235.23
eil76_3_9_1_n	0.12	0.24	0.12	0.24	0.20	0.36	1.43	6.14	6.69	2.73	2.92	2.69	2.96	482.14
eil76_3_9_4_n	0.16	0.37	0.16	0.37	0.16	0.37	0.00	6.82	7.17	8.60	10.04	8.62	9.92	278.73
kroA100_1_1_1_n	2.16	2.35	2.21	2.35	2.27	2.83	0.00	25.83	27.82	13.99	16.90	14.75	18.58	283.36
kroA100_1_1_4_n	2.07	2.53	1.23	1.42	1.23	1.42	0.00	30.91	33.89	70.24	83.44	70.05	81.84	123.87
kroA100_1_5_1_n	2.10	2.99	1.31	1.95	1.34	1.88	1.04	32.31	33.43	17.77	21.55	17.58	21.55	410.90
kroA100_1_5_4_n	0.59	0.62	0.13	0.13	0.13	0.13	0.00	22.03	23.76	40.45	45.49	40.12	45.59	160.72

Continued on next page

Table 6 – *Non-homogeneous case — Facility outsourcing — small instance — Continued from previous page*

	GRASP		GRASP+PR1		GRASP+PR2		SAA	GRASP		GRASP+PR1		GRASP+PR2		SAA
	avg	max	avg	max	avg	max	avg	avg	max	avg	max	avg	max	avg
kroA100_1_9_1_n	0.27	0.42	0.21	0.42	0.21	0.42	0.13	7.88	8.71	4.18	4.92	4.01	4.65	460.07
kroA100_1_9_4_n	1.16	1.47	0.28	0.70	0.25	0.70	0.72	12.96	14.08	18.54	24.64	18.83	24.53	197.00
kroA100_2_1_1_n	1.29	1.29	1.48	1.75	1.29	1.29	0.00	17.64	19.09	9.62	11.56	9.75	11.55	286.59
kroA100_2_1_4_n	1.79	1.79	0.00	0.00	0.00	0.00	0.30	20.54	21.78	8.64	9.78	8.55	9.62	122.15
kroA100_2_5_1_n	0.41	0.64	0.09	0.12	0.09	0.12	0.00	29.00	32.09	3.42	4.42	3.54	4.45	614.53
kroA100_2_5_4_n	0.73	1.72	0.72	1.72	0.72	1.72	0.04	14.75	16.28	29.83	37.67	29.23	35.77	150.27
kroA100_2_9_1_n	0.53	0.53	0.07	0.12	0.02	0.12	0.39	6.07	6.16	1.15	1.33	1.17	1.42	575.25
kroA100_2_9_4_n	0.11	0.11	0.00	0.00	0.00	0.00	0.91	25.70	27.73	13.45	16.39	13.43	18.27	147.87
kroA100_3_1_1_n	1.46	1.50	1.61	1.78	1.55	1.78	0.00	16.52	18.53	4.00	5.38	3.77	3.95	247.12
kroA100_3_1_4_n	0.58	0.58	0.53	0.58	0.58	0.58	0.00	25.54	26.60	19.64	20.98	19.38	22.20	122.67
kroA100_3_5_1_n	1.63	1.79	0.54	0.60	0.48	0.62	0.38	20.31	21.39	3.05	3.46	2.99	3.38	389.55
kroA100_3_5_4_n	0.46	0.46	0.16	0.45	0.34	0.88	0.00	13.37	14.42	7.26	8.46	7.22	8.40	162.18
kroA100_3_9_1_n	0.46	0.46	0.46	0.46	0.46	0.46	0.00	3.28	3.42	0.73	0.81	0.75	0.81	503.07
kroA100_3_9_4_n	0.79	0.79	0.07	0.26	0.06	0.14	1.76	11.08	11.36	3.94	4.32	4.18	4.72	212.59
kroB100_1_1_1_n	1.31	2.03	1.36	2.35	1.55	2.33	0.00	22.48	23.04	2.87	3.88	3.32	5.46	356.95
kroB100_1_1_4_n	3.80	3.94	3.65	3.94	3.80	3.94	0.00	29.99	33.85	25.95	28.58	26.25	31.00	147.89
kroB100_1_5_1_n	1.16	1.57	0.85	1.47	1.24	1.34	0.39	21.95	23.89	2.22	2.56	2.20	2.53	444.88
kroB100_1_5_4_n	5.52	6.22	1.29	6.08	1.29	6.08	0.00	13.62	14.41	5.82	7.16	5.96	7.18	153.06
kroB100_1_9_1_n	0.01	0.01	0.01	0.01	0.01	0.01	1.15	12.14	12.90	1.07	1.14	1.09	1.12	514.89
kroB100_1_9_4_n	0.59	0.59	0.00	0.00	0.00	0.00	0.07	14.77	15.44	9.26	10.73	9.45	10.84	156.28
kroB100_2_1_1_n	1.02	1.28	0.80	1.28	0.73	0.90	0.00	17.48	19.52	9.58	12.62	9.65	11.57	414.85
kroB100_2_1_4_n	0.56	1.98	0.12	0.49	0.07	0.23	0.19	18.68	20.71	21.53	24.69	21.79	25.02	203.35
kroB100_2_5_1_n	0.87	1.42	0.79	1.26	0.79	1.26	0.00	19.91	21.32	2.36	2.82	2.38	2.80	575.65
kroB100_2_5_4_n	1.64	1.64	0.59	1.01	0.48	1.00	1.04	9.37	10.36	19.80	25.48	19.04	25.40	249.44
kroB100_2_9_1_n	0.99	1.38	0.99	1.38	0.99	1.38	0.00	7.58	8.77	1.37	1.68	1.36	1.61	535.35
kroB100_2_9_4_n	0.77	1.05	0.60	1.05	0.54	0.74	0.00	4.67	5.17	11.39	13.92	11.87	14.35	190.94
kroB100_3_1_1_n	2.10	3.43	0.71	1.13	0.55	0.92	0.00	24.39	26.87	18.92	23.13	17.94	21.47	270.72
kroB100_3_1_4_n	1.35	1.70	0.85	1.45	0.85	1.49	0.00	23.34	24.94	33.24	39.44	33.13	40.29	151.17
kroB100_3_5_1_n	1.39	2.06	0.46	1.04	0.34	0.69	0.08	24.80	25.52	8.32	11.21	8.37	11.32	480.82
kroB100_3_5_4_n	0.66	0.92	0.15	0.25	0.15	0.25	0.02	20.50	21.91	30.30	38.70	30.29	38.85	146.28
kroB100_3_9_1_n	0.21	0.62	0.24	0.62	0.18	0.62	1.16	12.32	13.72	3.57	4.14	3.53	3.94	455.15
kroB100_3_9_4_n	1.64	2.29	1.18	1.62	0.75	0.92	0.00	6.78	7.67	13.19	13.58	13.01	13.41	163.26
kroC100_1_1_1_n	0.44	1.15	0.44	1.15	0.42	1.06	0.22	15.91	17.70	6.04	8.30	5.45	6.53	282.29
kroC100_1_1_4_n	2.40	2.92	0.17	0.87	0.17	0.87	0.03	20.24	24.91	75.95	84.20	71.87	78.97	114.32
kroC100_1_5_1_n	1.25	1.48	1.19	1.51	1.26	1.51	0.00	43.87	45.70	12.18	15.50	11.72	16.59	410.05
kroC100_1_5_4_n	1.52	2.00	0.04	0.07	0.36	1.00	0.33	17.43	19.69	23.31	31.98	23.25	32.73	122.18
kroC100_1_9_1_n	0.00	0.00	0.00	0.00	0.00	0.00	0.36	6.84	7.14	4.32	5.15	4.33	5.16	515.84
kroC100_1_9_4_n	0.03	0.17	0.03	0.17	0.03	0.17	0.34	18.90	21.44	17.18	20.60	17.82	20.05	168.15
kroC100_2_1_1_n	2.62	3.70	1.92	2.64	1.83	2.64	0.00	22.83	23.98	11.79	16.16	10.53	12.13	333.87
kroC100_2_1_4_n	6.36	6.52	2.90	5.18	5.48	5.73	0.00	21.09	22.86	15.63	21.29	21.53	22.40	127.66
kroC100_2_5_1_n	0.90	1.13	0.61	0.80	0.82	1.13	0.78	38.02	39.54	4.11	4.91	4.23	4.94	454.73
kroC100_2_5_4_n	0.02	0.04	0.03	0.04	0.04	0.04	0.42	23.76	24.47	19.62	20.52	19.66	20.42	150.64
kroC100_2_9_1_n	0.75	1.14	0.57	0.77	0.60	0.82	0.00	7.02	7.24	1.30	1.51	1.33	1.52	540.57
kroC100_2_9_4_n	0.00	0.00	0.00	0.00	0.08	0.19	0.75	18.01	18.75	6.80	7.36	6.81	7.36	191.29
kroC100_3_1_1_n	1.20	1.69	1.13	1.49	1.06	1.66	0.00	15.09	17.19	4.23	5.35	4.60	8.14	361.65
kroC100_3_1_4_n	0.00	0.00	0.00	0.00	0.00	0.00	0.62	25.74	27.20	7.01	9.73	6.65	11.96	164.53
kroC100_3_5_1_n	1.44	2.11	1.23	1.48	1.20	1.28	0.00	23.68	24.21	2.41	3.19	2.44	3.19	587.70
kroC100_3_5_4_n	2.13	2.13	1.83	2.13	1.83	2.13	0.00	15.09	16.43	23.40	29.32	23.45	29.25	205.17
kroC100_3_9_1_n	0.21	0.31	0.25	0.31	0.14	0.31	0.00	12.78	13.02	3.33	3.64	3.38	3.62	591.41
kroC100_3_9_4_n	1.31	3.00	1.19	2.43	1.17	2.31	0.00	7.94	9.00	17.39	23.02	17.27	22.93	199.79
kroD100_1_1_1_n	1.30	2.18	1.14	2.18	1.14	2.18	0.75	23.30	23.67	3.81	5.61	4.00	6.02	364.71
kroD100_1_1_4_n	2.02	3.86	0.73	1.36	0.55	1.20	0.93	21.34	22.36	33.04	37.88	33.12	36.70	201.73
kroD100_1_5_1_n	1.13	1.33	0.55	0.95	0.48	0.59	0.00	21.25	21.84	1.55	1.88	1.51	1.72	384.04
kroD100_1_5_4_n	0.94	1.28	0.03	0.15	0.09	0.45	0.56	17.31	19.06	27.32	29.34	26.93	28.41	180.79
kroD100_1_9_1_n	1.21	1.59	0.67	0.71	0.65	0.71	0.00	10.76	10.94	1.21	1.35	1.22	1.35	570.46
kroD100_1_9_4_n	1.33	1.74	0.26	0.46	0.26	0.46	0.95	14.47	15.42	19.97	20.80	19.98	20.92	222.16
kroD100_2_1_1_n	1.11	1.73	0.89	1.55	0.68	1.35	0.46	28.37	29.31	4.96	6.44	4.40	5.20	287.53
kroD100_2_1_4_n	0.56	1.04	0.42	0.44	0.42	0.44	0.00	27.66	28.36	19.98	25.18	20.40	26.41	132.82
kroD100_2_5_1_n	1.58	1.58	0.28	0.97	0.28	0.97	0.00	18.90	21.50	2.29	2.83	2.27	3.44	468.96
kroD100_2_5_4_n	0.16	0.16	0.00	0.02	0.00	0.00	0.07	10.92	11.91	17.89	22.11	17.76	23.16	147.03
kroD100_2_9_1_n	0.09	0.31	0.06	0.31	0.06	0.31	0.00	4.28	4.64	1.33	1.49	1.35	1.60	560.76

Continued on next page

Table 6 – *Non-homogeneous case — Facility outsourcing — small instance — Continued from previous page*

	GRASP		GRASP+PR1		GRASP+PR2		SAA	GRASP		GRASP+PR1		GRASP+PR2		SAA
	avg	max	avg	max	avg	max	avg	avg	max	avg	max	avg	max	avg
kroD100_2_9_4_n	0.08	0.08	0.08	0.08	0.08	0.08	0.00	13.22	13.88	6.45	7.13	6.85	7.27	224.74
kroD100_3_1_1_n	1.41	2.39	1.20	2.22	1.36	2.22	0.00	18.92	20.86	6.05	6.80	6.28	7.14	309.95
kroD100_3_1_4_n	0.71	0.71	0.57	0.71	0.57	0.71	0.22	20.88	23.17	8.02	10.12	7.80	9.50	151.14
kroD100_3_5_1_n	0.39	1.12	0.39	1.12	0.39	1.12	0.00	23.63	25.02	1.39	1.59	1.40	1.78	458.09
kroD100_3_5_4_n	1.76	1.76	0.38	0.38	0.38	0.38	0.00	16.54	16.80	9.71	10.17	9.25	9.89	156.12
kroD100_3_9_1_n	0.28	0.30	0.28	0.30	0.28	0.28	0.00	4.07	4.22	1.16	1.22	1.11	1.18	561.03
kroD100_3_9_4_n	0.19	0.19	0.19	0.19	0.19	0.19	0.00	9.09	9.91	7.16	8.52	7.09	8.33	198.96
kroE100_1_1_1_n	2.48	3.17	1.16	1.99	1.37	1.99	0.00	24.37	25.53	33.79	45.81	34.83	43.52	265.28
kroE100_1_1_4_n	0.47	1.56	0.47	1.56	0.47	1.56	0.00	20.79	23.31	30.00	34.53	30.15	36.07	116.49
kroE100_1_5_1_n	0.42	0.59	0.41	0.54	0.39	0.54	0.00	29.91	33.34	4.69	5.63	4.63	5.69	394.95
kroE100_1_5_4_n	0.43	1.20	0.41	1.20	0.41	1.20	0.48	12.56	13.26	29.04	33.81	27.61	33.69	139.76
kroE100_1_9_1_n	0.34	0.74	0.16	0.72	0.18	0.74	0.48	10.25	10.91	2.42	3.22	2.44	3.14	509.73
kroE100_1_9_4_n	1.91	1.91	0.34	0.34	0.27	0.34	0.41	15.67	16.61	21.21	22.61	21.15	23.40	161.34
kroE100_2_1_1_n	1.28	2.26	1.09	1.76	1.06	1.76	0.00	18.86	21.89	10.60	18.26	10.07	15.89	240.69
kroE100_2_1_4_n	2.53	5.45	1.90	4.63	1.90	4.63	1.35	30.29	33.03	41.53	63.89	40.69	63.71	120.60
kroE100_2_5_1_n	0.27	0.62	0.34	0.98	0.27	0.62	0.00	31.48	34.06	6.10	8.01	6.06	7.97	436.50
kroE100_2_5_4_n	0.75	1.03	0.72	1.03	0.72	1.03	0.00	10.79	12.14	11.44	13.46	11.87	15.03	120.78
kroE100_2_9_1_n	0.82	1.48	0.71	1.41	0.74	1.50	0.84	23.80	25.06	4.45	5.58	4.47	5.56	445.33
kroE100_2_9_4_n	1.26	2.73	1.00	2.75	1.00	2.73	0.00	10.31	10.81	5.47	7.64	5.64	7.48	150.73
kroE100_3_1_1_n	1.17	2.01	0.85	1.03	0.80	1.03	0.00	24.56	26.52	48.07	67.64	48.91	67.40	242.78
kroE100_3_1_4_n	0.00	0.00	0.00	0.00	0.00	0.00	1.52	23.64	24.84	24.56	28.22	24.63	27.41	133.85
kroE100_3_5_1_n	2.27	2.69	0.87	1.04	0.92	1.70	0.78	27.11	28.80	21.23	23.23	22.95	29.42	444.11
kroE100_3_5_4_n	0.76	0.76	0.00	0.00	0.00	0.01	0.21	21.44	22.39	40.64	44.18	38.89	45.66	181.30
kroE100_3_9_1_n	1.47	1.64	0.78	1.27	0.56	0.87	0.00	17.22	18.29	4.05	4.42	4.21	5.10	537.14
kroE100_3_9_4_n	0.91	0.91	0.33	0.91	0.41	0.91	0.00	17.93	18.53	13.12	15.45	12.16	14.23	169.54
pr76_1_1_1_n	0.64	1.07	0.81	0.97	0.71	1.18	0.00	26.52	28.19	16.88	20.37	18.04	22.39	426.12
pr76_1_1_4_n	2.67	2.72	1.57	1.68	1.52	1.68	0.00	27.97	28.91	38.90	47.52	37.75	45.84	171.78
pr76_1_5_1_n	1.48	1.52	0.91	1.12	0.82	1.12	0.00	25.62	26.82	5.71	7.05	5.76	6.65	452.26
pr76_1_5_4_n	0.07	0.07	0.07	0.07	0.07	0.07	0.00	6.63	7.28	17.63	19.58	21.13	22.79	145.92
pr76_1_9_1_n	0.42	0.47	0.36	0.41	0.37	0.47	0.00	5.40	5.61	1.53	1.70	1.44	1.76	520.93
pr76_1_9_4_n	1.21	1.24	0.00	0.00	0.00	0.00	0.58	17.78	18.38	15.49	16.88	15.42	16.27	204.58
pr76_2_1_1_n	1.86	1.86	0.81	1.03	1.14	1.59	0.00	20.01	20.83	7.21	9.83	7.03	8.87	387.18
pr76_2_1_4_n	1.61	1.61	0.04	0.22	0.00	0.00	0.41	32.70	34.61	61.90	72.71	58.00	72.28	174.58
pr76_2_5_1_n	1.70	1.94	0.78	0.84	1.12	2.08	0.00	23.39	25.25	1.53	1.77	1.65	1.94	545.86
pr76_2_5_4_n	2.91	5.13	0.02	0.08	1.10	1.10	0.41	16.26	17.27	4.98	5.77	3.97	4.74	205.93
pr76_2_9_1_n	0.03	0.07	0.01	0.07	0.01	0.07	3.46	8.78	9.69	1.24	1.40	1.26	1.41	470.54
pr76_2_9_4_n	0.07	0.07	0.07	0.07	0.07	0.07	0.00	3.63	3.75	6.20	6.44	6.12	6.53	222.37
pr76_3_1_1_n	0.68	1.00	0.57	1.08	0.47	1.20	0.00	18.80	19.64	3.05	3.69	3.01	3.33	381.82
pr76_3_1_4_n	0.57	0.57	0.17	0.23	0.13	0.21	0.56	25.50	27.00	35.85	42.28	34.43	41.73	201.08
pr76_3_5_1_n	1.01	1.29	0.01	0.03	0.02	0.03	1.09	8.14	8.81	0.97	1.07	1.05	1.13	456.17
pr76_3_5_4_n	1.65	3.12	0.00	0.00	0.00	0.00	0.70	12.13	12.68	11.03	14.23	10.97	13.49	218.08
pr76_3_9_1_n	0.56	0.80	0.21	0.33	0.21	0.26	0.00	2.05	2.16	0.96	1.02	0.94	1.02	477.13
pr76_3_9_4_n	0.80	0.80	0.16	0.16	0.13	0.16	0.07	4.28	4.54	5.39	6.12	6.38	6.74	261.13
rat99_1_1_1_n	0.67	1.00	0.67	1.00	0.67	1.00	0.30	19.20	21.84	5.90	7.01	5.62	6.83	310.21
rat99_1_1_4_n	1.29	1.29	0.51	0.62	0.62	0.62	0.00	21.10	22.58	16.57	21.55	16.90	21.19	155.93
rat99_1_5_1_n	2.26	3.45	2.28	3.45	2.28	3.39	0.00	25.29	26.60	1.83	2.31	1.67	2.19	654.61
rat99_1_5_4_n	0.15	0.15	0.06	0.15	0.06	0.15	0.23	13.94	14.47	10.82	13.47	10.68	12.75	148.20
rat99_1_9_1_n	0.60	1.09	0.58	1.00	0.61	1.10	0.49	18.94	19.75	1.63	1.94	1.67	2.06	596.39
rat99_1_9_4_n	2.01	3.13	0.61	1.13	0.55	1.08	0.39	10.28	11.69	7.67	9.59	7.24	9.49	242.38
rat99_2_1_1_n	1.85	2.39	1.49	1.82	1.44	1.80	0.00	20.56	22.80	12.79	17.44	15.70	22.25	325.52
rat99_2_1_4_n	4.16	4.51	1.57	1.97	1.57	1.97	0.13	20.79	21.80	17.38	19.69	17.79	21.10	161.05
rat99_2_5_1_n	1.59	1.73	0.48	0.97	0.66	1.16	1.00	19.80	21.47	5.84	7.02	5.95	7.33	369.15
rat99_2_5_4_n	1.17	1.78	0.55	0.55	0.51	0.55	0.00	20.55	21.53	39.13	42.73	38.79	40.82	143.56
rat99_2_9_1_n	1.02	1.39	0.08	0.24	0.14	0.56	0.63	14.20	14.85	1.99	2.08	1.92	2.05	512.19
rat99_2_9_4_n	1.90	2.57	0.03	0.08	0.03	0.08	0.57	9.50	10.13	21.07	25.89	21.39	27.50	200.02
rat99_3_1_1_n	2.77	2.99	1.80	2.99	1.74	2.70	0.00	17.89	18.86	9.73	10.19	8.40	9.89	288.43
rat99_3_1_4_n	1.59	1.84	1.86	2.55	1.98	2.55	0.00	14.14	15.56	21.86	28.09	20.08	25.64	116.46
rat99_3_5_1_n	0.62	1.12	0.48	0.70	0.36	0.70	0.00	24.77	26.64	14.17	15.47	13.88	15.52	368.72
rat99_3_5_4_n	2.02	2.02	0.01	0.07	0.04	0.07	0.63	19.12	20.10	24.22	30.95	24.43	28.10	118.18
rat99_3_9_1_n	1.13	1.14	0.68	0.94	0.81	0.94	0.18	7.15	7.33	6.01	7.23	6.21	7.28	469.99
rat99_3_9_4_n	0.30	0.58	0.20	0.22	0.20	0.22	0.00	10.24	11.00	24.26	25.73	24.28	25.63	177.68

Continued on next page

Table 6 – *Non-homogeneous case — Facility outsourcing — small instance — Continued from previous page*

	Percent gaps							Computing times						
	GRASP		GRASP+PR1		GRASP+PR2		SAA	GRASP		GRASP+PR1		GRASP+PR2		SAA
	avg	max	avg	max	avg	max	avg	avg	max	avg	max	avg	max	avg
st70_1_1_1_n	1.61	1.99	1.56	2.23	1.46	1.99	0.00	22.41	24.56	21.28	28.56	21.66	28.08	304.68
st70_1_1_4_n	1.88	1.88	1.43	1.43	1.43	1.43	0.00	33.14	36.45	53.92	55.47	52.70	56.10	144.88
st70_1_5_1_n	0.36	0.71	0.33	1.13	0.28	0.88	0.77	29.61	32.20	2.92	3.33	2.98	3.23	447.60
st70_1_5_4_n	0.71	0.71	0.47	0.47	0.47	0.47	0.00	29.36	30.00	25.39	27.97	24.83	27.73	200.65
st70_1_9_1_n	0.27	0.80	0.18	0.50	0.19	0.50	0.18	13.10	13.52	3.97	5.33	3.94	5.30	529.11
st70_1_9_4_n	0.04	0.04	0.12	0.47	0.11	0.47	0.09	11.20	11.96	5.45	6.73	5.45	6.39	214.73
st70_2_1_1_n	0.00	0.00	0.00	0.02	0.05	0.21	0.57	21.03	23.25	1.67	2.14	1.55	1.66	416.17
st70_2_1_4_n	0.87	0.87	0.80	0.87	0.87	0.89	0.00	25.82	27.59	6.39	7.14	6.09	7.57	172.07
st70_2_5_1_n	1.27	1.88	0.65	1.28	0.65	1.05	0.00	22.90	23.94	6.29	10.83	6.22	11.05	460.69
st70_2_5_4_n	1.36	1.36	0.29	1.06	0.73	1.31	0.00	5.90	6.19	7.60	8.38	8.82	11.54	193.02
st70_2_9_1_n	0.23	0.47	0.37	0.72	0.12	0.44	1.10	4.82	5.06	0.86	0.89	0.87	0.89	533.52
st70_2_9_4_n	0.19	0.19	0.00	0.00	0.04	0.19	0.02	10.65	11.46	9.80	11.73	10.01	12.18	229.69
st70_3_1_1_n	1.73	1.93	1.74	2.24	1.71	1.91	0.00	24.18	25.73	18.93	23.12	17.68	23.19	320.20
st70_3_1_4_n	7.39	7.39	5.91	7.39	7.39	7.39	0.25	33.73	35.38	22.03	27.95	23.21	27.28	159.71
st70_3_5_1_n	1.21	1.21	1.21	1.21	1.21	1.21	0.00	23.93	24.69	10.71	14.11	10.96	13.82	464.47
st70_3_5_4_n	0.89	0.89	0.39	0.45	0.32	0.45	0.00	13.95	14.38	25.86	29.50	25.54	29.19	179.66
st70_3_9_1_n	0.29	0.59	0.18	0.44	0.23	0.68	0.35	11.85	12.38	3.19	3.35	3.17	3.39	554.51
st70_3_9_4_n	1.51	1.51	0.68	1.05	0.63	1.05	0.43	9.86	10.22	11.33	13.06	11.31	13.85	219.59

Table 7: Non-homogeneous case — Facility outsourcing — large instances.

	Percent gaps							Computing times						
	GRASP		GRASP+PR1		GRASP+PR2		SAA	GRASP		GRASP+PR1		GRASP+PR2		SAA
	avg	max	avg	max	avg	max	avg	avg	max	avg	max	avg	max	avg
LkroA100_1_1_1_n	1.69	2.33	1.69	2.33	1.69	2.33	0.00	17.59	18.58	5.80	7.22	5.75	7.39	3278.88
LkroA100_1_1_4_n	0.25	0.64	0.03	0.09	0.03	0.09	0.75	27.86	31.86	82.90	93.70	86.36	93.59	643.83
LkroA100_1_5_1_n	1.10	1.63	0.52	0.95	0.42	0.71	0.00	12.44	12.91	3.32	4.28	3.34	3.99	2592.66
LkroA100_1_5_4_n	0.03	0.03	0.03	0.03	0.02	0.03	0.11	16.19	16.69	55.38	83.52	54.40	79.74	702.93
LkroA100_1_9_1_n	1.41	1.95	0.52	1.03	0.52	0.97	0.81	11.79	13.01	2.28	2.36	2.27	2.39	1867.62
LkroA100_1_9_4_n	1.59	2.06	1.02	1.35	1.15	1.93	0.00	6.93	7.72	24.32	26.89	23.58	25.74	841.98
LkroA100_2_1_1_n	2.42	3.14	1.46	2.44	1.47	2.44	0.00	14.63	14.90	6.41	7.76	6.37	7.59	1841.93
LkroA100_2_1_4_n	1.24	1.65	0.63	1.25	0.63	1.25	0.00	22.12	23.89	85.53	95.15	86.20	96.47	547.49
LkroA100_2_5_1_n	1.87	2.19	0.44	0.79	0.39	0.75	0.01	22.82	24.61	9.39	14.06	9.76	14.30	1409.38
LkroA100_2_5_4_n	2.58	2.96	0.79	2.15	0.85	2.15	0.42	13.99	15.30	66.56	83.73	66.28	83.96	645.66
LkroA100_2_9_1_n	0.35	0.73	0.35	0.73	0.35	0.73	0.50	18.86	20.38	2.82	3.09	2.86	3.05	1681.89
LkroA100_2_9_4_n	1.20	1.76	0.42	1.04	0.42	1.04	0.00	13.16	14.85	25.62	35.93	26.05	37.43	887.90
LkroA100_3_1_1_n	2.71	3.15	2.59	3.18	2.64	3.14	0.00	15.01	16.33	2.84	3.54	2.92	3.54	2431.68
LkroA100_3_1_4_n	2.09	2.92	2.19	2.92	1.93	2.92	0.00	14.58	15.96	8.00	11.59	8.34	10.69	684.68
LkroA100_3_5_1_n	1.65	2.38	0.87	1.41	0.85	1.32	0.00	16.96	18.00	4.49	5.07	4.49	5.13	1587.61
LkroA100_3_5_4_n	1.83	2.19	0.15	0.26	0.13	0.26	1.06	16.46	18.77	56.40	59.45	56.83	63.18	861.81
LkroA100_3_9_1_n	1.72	2.27	1.35	1.59	1.37	1.59	0.00	14.50	15.48	2.60	2.88	2.59	2.83	1941.49
LkroA100_3_9_4_n	2.32	2.53	0.04	0.09	0.08	0.23	1.82	7.66	8.27	18.48	25.44	18.05	25.13	830.75
LkroB100_1_1_1_n	1.16	1.45	1.11	1.45	0.98	1.45	0.00	17.55	19.86	9.46	10.68	9.28	11.29	2778.91
LkroB100_1_1_4_n	1.57	1.94	0.21	0.44	0.21	0.44	0.00	23.04	25.02	76.41	99.65	76.22	102.57	553.42
LkroB100_1_5_1_n	0.87	1.69	0.85	1.59	0.97	1.59	0.00	18.12	19.61	5.16	6.69	5.06	6.40	5034.00
LkroB100_1_5_4_n	0.56	0.79	0.22	0.53	0.22	0.53	0.28	15.26	16.38	56.25	62.54	56.35	62.09	616.14
LkroB100_1_9_1_n	0.55	0.74	0.48	0.70	0.51	0.74	0.00	20.64	21.30	6.41	7.31	6.48	7.63	1872.38
LkroB100_1_9_4_n	1.13	1.52	0.45	1.21	0.46	1.21	0.69	6.65	7.22	21.60	28.74	21.86	28.45	938.80
LkroB100_2_1_1_n	1.64	2.14	1.74	2.08	1.85	2.50	0.00	10.72	11.71	2.14	2.33	2.13	2.48	2770.65
LkroB100_2_1_4_n	0.46	0.80	0.38	0.80	0.35	0.80	1.01	17.92	20.17	31.58	38.83	25.69	37.55	838.79
LkroB100_2_5_1_n	1.29	1.51	0.88	1.24	0.81	1.24	0.00	12.18	13.01	2.59	2.70	2.61	2.69	2422.38
LkroB100_2_5_4_n	0.63	0.81	0.15	0.51	0.15	0.63	3.18	14.96	16.14	21.90	32.50	21.53	32.31	624.22
LkroB100_2_9_1_n	0.60	0.90	0.41	0.70	0.38	0.59	0.17	9.60	10.31	2.51	2.66	2.50	2.62	1702.19
LkroB100_2_9_4_n	2.01	2.82	0.38	0.88	0.42	0.88	0.98	7.89	8.38	25.07	32.17	24.61	32.35	818.41
LkroB100_3_1_1_n	1.79	2.31	1.79	2.31	1.85	2.31	0.00	14.31	15.18	3.40	3.80	3.38	3.87	2190.11
LkroB100_3_1_4_n	0.34	0.83	0.34	0.83	0.34	0.83	0.48	22.27	23.27	44.06	50.61	43.84	50.37	787.56
LkroB100_3_5_1_n	0.61	0.95	0.27	0.65	0.27	0.65	0.11	32.46	33.96	11.35	13.84	10.88	13.46	1983.17

Continued on next page

Table 7 – *Non-homogeneous case — Facility outsourcing — small instance — Continued from previous page*

	GRASP		GRASP+PR1		GRASP+PR2		SAA	GRASP		GRASP+PR1		GRASP+PR2		SAA
	avg	max	avg	max	avg	max	avg	avg	max	avg	max	avg	max	avg
LkroB100_3_5_4_n	1.29	1.88	1.18	2.48	1.15	2.33	0.55	28.16	29.64	78.97	103.16	78.68	103.25	693.63
LkroB100_3_9_1_n	0.60	1.17	0.53	1.00	0.61	1.00	0.46	11.44	11.94	2.28	2.41	2.28	2.42	1966.84
LkroB100_3_9_4_n	0.43	0.57	0.40	0.46	0.34	0.52	2.63	13.87	14.37	45.67	54.79	44.17	59.03	921.10
LkroC100_1_1_1_n	1.75	1.97	1.67	1.97	1.67	2.19	0.00	15.10	16.80	4.36	5.33	4.35	5.33	3043.67
LkroC100_1_1_4_n	0.97	1.25	0.91	1.25	0.92	1.25	0.00	23.86	26.25	141.61	160.02	141.71	159.98	514.48
LkroC100_1_5_1_n	0.36	0.62	0.40	0.66	0.38	0.63	0.12	14.63	15.36	2.62	2.81	2.66	2.97	5340.08
LkroC100_1_5_4_n	3.30	3.42	0.50	0.64	0.41	0.64	0.57	17.37	18.11	105.51	132.61	106.64	126.52	721.03
LkroC100_1_9_1_n	0.35	1.03	0.35	1.03	0.33	0.93	0.10	20.84	21.52	2.11	2.27	2.11	2.25	2066.34
LkroC100_1_9_4_n	0.45	1.80	0.45	1.82	0.22	0.69	0.87	8.94	9.92	14.57	24.08	13.57	23.82	942.67
LkroC100_2_1_1_n	2.10	2.80	2.12	2.80	2.16	2.97	0.00	16.36	18.05	5.71	6.39	5.54	6.50	5130.06
LkroC100_2_1_4_n	2.53	3.04	0.59	1.66	0.59	1.66	0.57	15.38	16.04	37.82	50.60	36.36	49.66	1058.05
LkroC100_2_5_1_n	1.31	1.62	0.47	0.75	0.41	0.78	0.00	23.41	24.46	3.63	3.86	3.60	3.83	5843.53
LkroC100_2_5_4_n	2.83	3.70	0.18	0.46	0.26	0.43	1.01	25.69	27.72	81.92	97.59	82.57	99.12	722.84
LkroC100_2_9_1_n	1.73	2.14	1.70	1.85	1.69	1.97	0.00	19.58	20.25	3.04	3.50	3.08	3.44	2334.59
LkroC100_2_9_4_n	2.32	2.42	1.36	2.27	1.38	2.27	0.18	13.29	14.14	12.36	15.33	12.53	14.58	787.01
LkroC100_3_1_1_n	1.18	1.61	1.18	1.61	1.26	1.73	0.00	15.17	16.13	3.78	4.32	3.61	4.16	3437.77
LkroC100_3_1_4_n	1.89	2.54	1.72	2.44	1.68	2.44	0.00	17.57	18.52	23.58	33.46	22.66	33.39	862.13
LkroC100_3_5_1_n	1.30	1.63	0.80	1.14	0.87	1.43	0.16	8.01	9.20	2.79	2.86	2.82	2.89	2513.94
LkroC100_3_5_4_n	2.05	2.30	0.12	0.12	0.09	0.12	0.51	19.88	21.44	57.78	65.30	57.21	64.85	616.86
LkroC100_3_9_1_n	1.11	1.69	0.65	1.08	0.78	1.08	1.65	16.14	16.67	2.41	2.57	2.50	2.70	2148.79
LkroC100_3_9_4_n	2.61	3.56	0.79	1.77	0.79	1.92	0.19	6.23	7.03	20.77	22.77	20.47	25.34	828.55
LkroD100_1_1_1_n	2.16	3.08	1.51	2.40	1.43	2.40	0.00	11.40	12.24	3.19	3.41	3.25	3.52	3403.21
LkroD100_1_1_4_n	0.73	0.98	0.64	0.98	0.36	0.98	0.18	10.83	12.02	30.01	36.06	28.60	35.85	919.39
LkroD100_1_5_1_n	3.20	3.76	0.66	1.33	0.79	1.67	0.22	18.57	20.11	5.37	5.91	5.99	6.80	1585.46
LkroD100_1_5_4_n	2.17	4.06	1.03	1.89	1.05	1.80	0.30	11.25	11.62	24.56	30.35	26.91	34.36	847.41
LkroD100_1_9_1_n	0.56	0.72	0.19	0.19	0.27	0.48	0.00	8.86	9.00	2.58	2.66	2.59	2.72	1710.02
LkroD100_1_9_4_n	2.35	2.88	0.12	0.14	0.14	0.28	1.42	8.87	10.03	16.74	21.03	16.69	20.79	812.18
LkroD100_2_1_1_n	2.44	3.09	2.00	2.43	2.08	2.48	0.00	14.07	14.53	3.21	4.21	3.00	3.34	1739.17
LkroD100_2_1_4_n	0.98	1.19	0.38	0.70	0.30	0.70	0.00	15.95	17.55	35.74	44.19	35.29	43.95	792.00
LkroD100_2_5_1_n	2.27	2.87	0.88	1.77	0.83	1.77	1.74	29.52	30.09	3.43	4.26	3.45	4.53	1546.24
LkroD100_2_5_4_n	2.48	2.94	0.47	1.27	0.47	1.27	0.63	26.16	28.88	41.19	63.41	41.15	62.85	798.67
LkroD100_2_9_1_n	0.53	0.98	0.48	1.13	0.41	0.83	0.94	9.12	9.48	2.31	2.48	2.30	2.48	1814.16
LkroD100_2_9_4_n	1.49	2.61	0.66	1.16	0.66	1.16	0.51	10.54	10.77	20.68	24.83	20.36	25.06	705.07
LkroD100_3_1_1_n	1.69	2.55	1.48	2.22	1.69	2.31	0.00	11.86	12.66	3.08	3.64	2.92	3.52	4107.31
LkroD100_3_1_4_n	1.64	1.76	0.81	2.09	0.79	1.68	0.16	14.21	15.05	40.02	52.44	35.02	48.22	771.62
LkroD100_3_5_1_n	1.02	1.26	1.01	1.19	0.97	1.19	0.00	16.67	18.03	2.99	3.20	2.96	3.00	2286.23
LkroD100_3_5_4_n	1.86	1.86	0.11	0.15	0.14	0.45	0.97	17.05	17.83	47.22	50.55	47.23	50.91	900.89
LkroD100_3_9_1_n	1.04	1.39	0.25	0.59	0.26	0.59	0.42	15.36	17.09	2.52	2.63	2.55	2.72	1963.83
LkroD100_3_9_4_n	1.83	1.96	0.06	0.23	0.02	0.09	1.66	9.73	10.23	47.65	51.06	47.23	50.16	859.19
LkroE100_1_1_1_n	0.80	1.75	0.77	1.62	0.77	1.62	0.00	18.52	19.85	3.05	3.54	3.13	3.61	3058.29
LkroE100_1_1_4_n	1.16	1.66	0.95	1.53	0.95	1.53	0.45	20.52	22.16	24.36	28.94	24.39	28.90	828.17
LkroE100_1_5_1_n	0.89	1.10	0.73	1.10	0.68	1.09	0.00	6.80	7.53	2.88	3.16	2.94	3.30	2258.17
LkroE100_1_5_4_n	1.32	2.40	1.10	1.33	1.16	1.33	0.00	16.07	17.97	28.18	32.72	28.43	34.18	823.02
LkroE100_1_9_1_n	1.08	1.32	0.34	0.68	0.33	0.78	0.49	7.16	8.52	2.69	2.77	2.67	2.72	1722.24
LkroE100_1_9_4_n	0.82	1.19	0.02	0.12	0.02	0.12	0.72	10.44	11.23	29.58	33.87	29.71	33.90	956.13
LkroE100_2_1_1_n	1.20	1.78	1.15	1.78	1.33	1.90	0.00	20.47	22.11	2.68	3.11	2.68	3.09	4782.04
LkroE100_2_1_4_n	2.23	2.71	1.99	2.56	2.11	3.16	0.00	21.29	23.42	12.32	20.14	12.22	19.66	651.00
LkroE100_2_5_1_n	1.57	2.63	1.11	1.53	1.04	1.41	0.00	19.93	20.56	2.74	3.02	2.71	2.86	5691.25
LkroE100_2_5_4_n	3.76	4.23	1.72	3.04	1.72	3.04	0.78	20.31	20.78	61.90	90.82	62.28	91.28	739.98
LkroE100_2_9_1_n	1.17	1.75	0.65	1.19	0.65	1.47	0.08	24.49	25.14	3.05	4.22	3.13	4.84	2583.16
LkroE100_2_9_4_n	0.44	0.64	0.44	0.64	0.44	0.64	0.00	11.96	12.44	18.21	23.75	21.72	32.21	967.16
LkroE100_3_1_1_n	2.71	3.20	2.22	3.02	2.21	3.16	0.00	16.16	17.72	11.06	16.31	10.98	16.53	2397.99
LkroE100_3_1_4_n	1.58	1.84	0.96	1.45	1.18	1.59	0.00	19.54	21.16	65.83	80.12	64.95	80.22	674.51
LkroE100_3_5_1_n	1.63	1.98	0.95	1.06	0.74	1.12	0.00	16.81	17.97	6.27	7.46	6.35	7.90	4048.27
LkroE100_3_5_4_n	3.13	3.62	0.75	1.48	0.83	1.48	3.03	11.48	12.40	58.28	80.16	59.30	81.86	681.51
LkroE100_3_9_1_n	1.22	1.34	1.02	1.30	1.03	1.30	0.00	21.32	22.60	5.64	7.04	5.47	6.18	1472.62
LkroE100_3_9_4_n	0.86	1.50	0.06	0.14	0.08	0.17	1.98	9.36	10.10	28.80	34.53	28.68	35.00	680.67
Lrat99_1_1_1_n	1.85	2.09	1.86	2.09	1.85	2.09	0.00	14.30	15.03	3.35	5.25	3.36	5.03	3562.83
Lrat99_1_1_4_n	2.04	2.41	0.39	1.40	0.19	0.35	0.87	14.21	15.19	67.93	82.62	75.38	82.27	839.65
Lrat99_1_5_1_n	3.29	3.63	1.17	2.00	1.17	2.00	0.60	24.90	25.92	18.67	25.02	18.58	25.10	1666.21

Continued on next page

Table 7 – *Non-homogeneous case — Facility outsourcing — small instance — Continued from previous page*

	GRASP		GRASP+PR1		GRASP+PR2		SAA	GRASP		GRASP+PR1		GRASP+PR2		SAA
	avg	max	avg	max	avg	max	avg	avg	max	avg	max	avg	max	avg
Lrat99_1_5_4_n	1.76	1.91	0.08	0.19	0.08	0.19	0.76	21.77	22.70	45.31	54.12	45.48	54.77	682.31
Lrat99_1_9_1_n	0.51	0.88	0.46	0.73	0.46	0.78	0.27	21.45	23.41	2.46	2.59	2.50	2.58	1719.44
Lrat99_1_9_4_n	1.68	1.94	0.92	1.23	0.89	1.23	0.43	4.14	4.61	46.14	52.39	45.68	52.55	762.78
Lrat99_2_1_1_n	2.37	2.49	2.43	2.78	2.19	2.78	0.00	20.39	21.03	5.50	9.46	5.53	8.69	4870.12
Lrat99_2_1_4_n	1.14	3.60	0.61	1.87	0.62	1.91	0.03	23.96	24.77	27.63	34.54	25.67	32.80	938.21
Lrat99_2_5_1_n	0.78	1.24	0.72	0.91	0.81	1.39	0.00	20.72	21.94	2.31	2.56	2.31	2.50	2700.29
Lrat99_2_5_4_n	0.48	1.12	0.23	0.82	0.23	0.82	0.40	18.61	19.32	30.52	41.21	29.53	41.69	879.20
Lrat99_2_9_1_n	1.16	1.52	0.54	0.91	0.56	1.11	0.68	15.62	16.98	2.45	2.62	2.53	2.80	2103.34
Lrat99_2_9_4_n	0.71	0.72	0.43	0.62	0.34	0.71	2.32	5.55	5.77	36.48	41.89	34.57	40.91	901.90
Lrat99_3_1_1_n	2.41	3.16	2.41	3.16	2.40	3.16	0.00	17.39	18.11	2.09	2.20	2.04	2.20	2187.23
Lrat99_3_1_4_n	1.97	3.81	1.28	2.35	1.59	2.35	0.84	19.49	20.84	21.82	30.59	20.42	30.64	787.41
Lrat99_3_5_1_n	2.09	2.62	0.82	1.69	0.61	1.01	0.41	18.37	19.51	2.66	2.88	2.62	2.77	2261.62
Lrat99_3_5_4_n	2.53	2.72	0.61	1.57	0.62	1.57	0.11	16.57	17.77	41.08	66.17	41.23	60.17	914.36
Lrat99_3_9_1_n	0.57	0.93	0.34	0.70	0.28	0.70	0.72	12.27	13.88	2.39	2.64	2.41	2.63	1855.90
Lrat99_3_9_4_n	0.13	0.15	0.09	0.13	0.09	0.13	0.85	5.29	5.65	32.24	35.72	31.95	35.73	854.88

Table 8: Non-homogeneous case — Customer outsourcing — small instances.

	Percent gaps							Computing times						
	GRASP		GRASP+PR1		GRASP+PR2		SAA	GRASP		GRASP+PR1		GRASP+PR2		SAA
	avg	max	avg	max	avg	max	avg	avg	max	avg	max	avg	max	avg
berlin52_1_1_1_n	1.49	2.75	1.27	2.62	1.27	2.61	1.37	2.51	2.81	1.15	1.26	1.16	1.36	31643.27
berlin52_1_1_4_n	0.89	1.04	0.62	1.28	0.69	1.04	2.19	1.12	1.16	0.97	1.07	0.92	0.95	4236.97
berlin52_1_5_1_n	1.84	2.01	1.25	1.99	1.13	1.88	1.76	2.00	2.20	1.14	1.22	1.14	1.21	50331.34
berlin52_1_5_4_n	0.71	0.71	0.37	0.86	0.74	1.30	1.49	1.02	1.08	0.99	1.08	1.01	1.18	26335.80
berlin52_1_9_1_n	0.45	0.83	0.45	0.83	0.38	0.80	1.19	0.87	0.91	0.96	1.00	0.88	0.92	87818.44
berlin52_1_9_4_n	0.72	1.01	0.44	1.02	0.40	1.02	3.46	0.67	0.70	0.93	0.96	0.91	1.03	7325.74
eil51_1_1_1_n	0.39	0.49	0.38	0.94	0.39	0.49	3.31	0.86	0.91	1.08	1.18	1.08	1.14	26017.56
eil51_1_1_4_n	2.95	3.10	0.21	1.05	2.17	2.24	0.72	0.97	1.03	0.94	1.05	0.89	0.95	5756.55
eil51_1_5_1_n	1.22	1.64	1.02	1.81	1.14	1.95	2.51	1.00	1.05	1.08	1.15	1.06	1.21	97438.22
eil51_1_5_4_n	0.00	0.00	0.00	0.00	0.00	0.00	1.97	0.69	0.72	1.00	1.19	1.01	1.08	15021.36
eil51_1_9_1_n	1.08	2.19	0.41	1.30	0.42	1.14	2.64	0.87	0.91	0.93	1.02	0.91	1.05	10658.00
eil51_1_9_4_n	3.45	4.32	0.00	0.00	0.00	0.00	6.93	0.67	0.69	0.86	0.91	0.81	0.93	8933.70
eil76_1_1_1_n	2.22	4.29	0.31	1.25	0.84	2.40	2.07	0.94	0.95	1.20	1.33	1.23	1.34	38476.67
eil76_1_1_4_n	2.18	2.18	2.18	2.18	2.18	2.18	0.00	0.79	0.81	0.87	0.92	0.86	0.91	7013.58
eil76_1_5_1_n	0.35	0.59	0.29	0.72	0.41	0.94	0.54	1.12	1.16	1.21	1.28	1.19	1.24	70026.88
eil76_1_5_4_n	0.05	0.06	0.14	0.35	0.08	0.35	0.27	0.70	0.73	0.94	1.02	0.93	1.08	14561.11
eil76_1_9_1_n	0.13	0.33	0.07	0.33	0.00	0.00	0.63	1.23	1.31	1.09	1.13	1.07	1.11	66291.17
eil76_1_9_4_n	0.31	1.13	0.03	0.11	0.03	0.03	1.64	0.75	0.78	1.03	1.11	1.02	1.08	10336.49
kroA100_1_1_1_n	0.51	1.27	0.61	1.77	0.51	1.27	3.82	2.57	2.83	1.00	1.11	1.06	1.13	24919.27
kroA100_1_1_4_n	1.23	1.72	0.38	0.65	0.54	1.06	1.20	0.97	0.99	2.62	3.06	2.60	2.98	6956.27
kroA100_1_5_1_n	1.37	2.20	0.89	1.40	1.00	1.80	1.80	1.23	1.30	1.67	1.80	1.67	1.78	109627.97
kroA100_1_5_4_n	0.96	1.44	0.47	0.80	0.37	0.77	3.79	0.83	0.89	0.93	1.03	0.98	1.11	15284.99
kroA100_1_9_1_n	1.17	1.71	1.18	1.71	1.17	1.71	0.00	0.97	1.00	1.88	2.13	1.92	2.17	76521.17
kroA100_1_9_4_n	1.37	1.37	0.88	1.64	1.01	1.37	1.20	0.78	0.84	0.88	0.98	0.89	1.02	20995.44
kroB100_1_1_1_n	0.78	1.73	0.40	0.62	0.41	0.62	0.44	1.03	1.13	0.95	0.99	0.94	0.99	23952.41
kroB100_1_1_4_n	2.31	4.19	2.58	5.56	2.31	4.19	0.00	0.95	1.01	0.85	0.93	0.82	0.92	5193.89
kroB100_1_5_1_n	0.77	1.38	0.47	1.22	0.44	0.63	3.19	1.00	1.09	1.56	1.66	1.46	1.52	80178.92
kroB100_1_5_4_n	3.16	3.82	1.82	2.77	1.74	2.75	1.88	0.77	0.83	0.77	0.82	0.77	0.83	26318.86
kroB100_1_9_1_n	0.32	0.57	0.14	0.47	0.12	0.24	0.21	2.16	2.61	0.99	1.03	0.98	1.03	94085.58
kroB100_1_9_4_n	1.62	1.62	0.09	0.43	0.09	0.43	1.07	0.79	0.84	1.10	1.21	1.08	1.25	18683.25
kroC100_1_1_1_n	0.31	0.97	0.40	0.78	0.17	0.60	2.68	0.97	1.05	1.32	1.47	1.32	1.41	14273.67

Continued on next page

Table 8 – *Non-homogeneous case — Facility outsourcing — small instance — Continued from previous page*

	GRASP		GRASP+PR1		GRASP+PR2		SAA	GRASP		GRASP+PR1		GRASP+PR2		SAA
	avg	max	avg	max	avg	max	avg	avg	max	avg	max	avg	max	avg
kroC100_1_1_4_n	1.33	2.52	0.62	2.12	0.50	2.52	2.88	0.74	0.78	1.29	1.47	1.27	1.45	3986.08
kroC100_1_5_1_n	0.45	2.26	0.61	1.95	0.28	1.40	3.42	1.31	1.41	1.48	2.03	1.39	1.63	94485.06
kroC100_1_5_4_n	3.17	3.47	0.74	1.45	1.08	1.58	5.31	0.87	0.95	0.93	1.05	0.96	1.08	12873.92
kroC100_1_9_1_n	0.41	1.21	0.37	1.01	0.37	1.01	2.53	0.87	0.91	1.20	1.31	1.20	1.34	103566.05
kroC100_1_9_4_n	0.22	0.45	0.13	0.43	0.13	0.43	2.21	0.71	0.75	1.11	1.22	1.06	1.14	9007.88
kroD100_1_1_1_n	2.41	3.85	2.14	3.21	1.99	3.11	3.39	2.19	2.47	1.08	1.24	1.01	1.11	29718.97
kroD100_1_1_4_n	2.60	3.20	0.47	1.29	0.48	1.31	1.53	1.10	1.14	1.01	1.06	1.02	1.16	6279.09
kroD100_1_5_1_n	0.90	1.11	0.59	0.84	0.53	0.88	1.35	1.28	1.34	1.10	1.16	1.12	1.22	77163.94
kroD100_1_5_4_n	0.56	0.60	0.34	0.87	0.29	0.61	0.76	0.77	0.81	0.80	0.88	0.78	0.84	14585.13
kroD100_1_9_1_n	0.38	0.98	0.31	0.62	0.42	0.73	4.07	0.96	0.98	1.13	1.24	1.14	1.25	7377.02
kroD100_1_9_4_n	1.10	1.54	0.15	0.45	0.21	0.45	2.13	0.65	0.69	1.06	1.22	1.05	1.25	8776.03
kroE100_1_1_1_n	1.06	2.27	0.54	1.00	0.38	1.00	1.89	0.93	0.98	1.78	2.19	1.82	2.27	20631.14
kroE100_1_1_4_n	0.17	0.28	0.28	0.83	0.17	0.28	1.06	0.82	0.84	0.88	0.94	0.87	0.96	4429.39
kroE100_1_5_1_n	0.30	1.00	0.30	1.00	0.47	1.00	2.67	1.89	1.92	1.46	1.85	1.43	1.74	63229.84
kroE100_1_5_4_n	0.47	0.73	0.02	0.06	0.01	0.06	4.10	0.69	0.70	0.95	1.00	0.93	0.97	14158.50
kroE100_1_9_1_n	0.59	0.89	0.56	0.87	0.59	0.89	0.00	1.22	1.28	1.14	1.23	1.16	1.19	69202.53
kroE100_1_9_4_n	2.25	2.25	0.84	2.25	0.84	2.25	2.14	0.93	1.06	1.08	1.13	1.06	1.14	8199.02
pr76_1_1_1_n	0.15	0.50	0.09	0.23	0.19	0.23	1.75	1.41	1.50	1.53	1.63	1.49	1.61	25683.59
pr76_1_1_4_n	3.33	3.97	1.35	1.77	1.38	2.00	1.83	0.95	1.01	1.13	1.26	1.16	1.27	4649.45
pr76_1_5_1_n	0.00	0.00	0.28	0.62	0.16	0.39	2.10	1.19	1.30	1.29	1.47	1.24	1.45	136427.13
pr76_1_5_4_n	0.45	0.67	0.15	0.44	0.23	0.64	0.28	0.71	0.73	0.92	1.02	0.94	0.99	11323.27
pr76_1_9_1_n	0.35	0.51	0.12	0.59	0.16	0.78	2.86	0.81	0.84	0.92	0.94	0.94	1.03	16642.84
pr76_1_9_4_n	0.92	1.29	0.24	0.83	0.29	0.93	1.61	0.69	0.72	0.77	0.80	0.80	0.86	29080.86
rat99_1_1_1_n	3.00	3.82	2.74	3.71	2.00	3.29	2.47	1.31	1.39	1.16	1.25	1.19	1.31	17285.69
rat99_1_1_4_n	1.03	2.65	0.11	0.23	0.08	0.23	1.11	0.96	0.99	1.18	1.41	1.19	1.43	6542.63
rat99_1_5_1_n	0.15	0.27	0.11	0.27	0.10	0.27	0.11	0.96	0.97	1.05	1.11	1.03	1.11	87151.74
rat99_1_5_4_n	0.46	0.46	0.00	0.00	0.18	0.46	0.81	0.79	0.85	0.90	0.93	0.85	0.91	15986.58
rat99_1_9_1_n	0.62	1.44	0.62	0.98	0.46	0.91	2.92	1.20	1.28	0.95	1.01	0.95	0.97	8223.08
rat99_1_9_4_n	2.02	2.86	1.40	2.05	1.00	1.80	3.08	1.03	1.14	1.01	1.10	0.99	1.05	30016.55
st70_1_1_1_n	1.23	2.25	0.76	1.46	0.76	1.46	1.63	1.21	1.25	1.24	1.44	1.19	1.39	20808.30
st70_1_1_4_n	0.00	0.00	0.00	0.00	0.09	0.47	1.93	1.03	1.06	1.15	1.41	1.12	1.38	6354.81
st70_1_5_1_n	0.45	0.93	0.33	0.58	0.29	0.58	1.61	1.41	1.56	1.37	1.41	1.36	1.42	175558.63
st70_1_5_4_n	0.65	1.12	0.41	1.12	0.63	1.28	3.80	0.91	1.00	0.77	0.84	0.75	0.78	16933.95
st70_1_9_1_n	0.59	1.04	0.59	1.04	0.63	1.24	2.04	1.11	1.17	0.94	0.97	0.96	0.99	51513.94
st70_1_9_4_n	0.77	0.77	0.31	0.77	0.61	0.77	1.59	0.75	0.77	0.79	0.83	0.80	0.88	6083.91