Optimal shift partitioning of pharmacies

Giovanni Andreatta, Luigi De Giovanni, University of Padova, Department of Mathematics
Paolo Serafini, University of Udine, Department of Mathematics and Computer Science

Abstract. The pharmacy service requires that some pharmacies are always available and shifts have to be organized: a shift corresponds to a subset of pharmacies that must be open 24 hours a day on a particular week. Under the requirement that each pharmacy belongs to exactly one shift and the assumption that users minimize the distance to the closest open pharmacy during each shift, we want to determine a partition of the pharmacies into a given number of shifts, such that the total distance covered by users is minimized. It may be also required that shift cardinalities are balanced. We discuss different versions and the related computational complexity, showing that the problem is NP-Hard in general. A set packing formulation is presented and solved by branch-and-price, together with a fast solution technique based on a tabu search. They have been applied to real and random instances showing that (i) the set packing formulation is very tight and often exhibits no integrality gap; (ii) the branch-and-price solves problems of practical relevance to optimality in a reasonable amount of time (order of minutes); (iii) the tabu search finds optimal or near-optimal solutions in order of seconds.

1 Introduction

Pharmacies in Italy are privately owned and managed. However, due to the particular service they have to provide, pharmacies are subject to special rules that usual shops do not have to obey. For instance, opening a new pharmacy can be allowed only if the ratio pharmacy/population is below a certain threshold and if a new pharmacy can be opened, municipalities are requested to suggest the most convenient location for the community. Moreover, people must be granted that some pharmacies, not too far away, are always available for service, day and night and also on holidays.

To meet this request, pharmacies on large territorial units mutually agree to establish shifts of duty on a rotational basis. A shift lasts a week and during this week the pharmacy must be open 24 hours (although for safety reasons they are locked at night and let customers in only on motivated request). Once all pharmacies have carried out their duty the same shift pattern is repeated.

Essentially the problem is to decide which pharmacies have to carry out their duty in the same week with the idea that these pharmacies must be sufficiently spread over the territory. As already remarked, the pharmacy locations are fixed and cannot be changed. The number of shifts is also usually fixed in advance, even if in principle it could change. Clearly a higher number of shifts corresponds to a lighter burden for the pharmacies but to a poorer service to the community.

In mathematical terms one has to partition the set of pharmacies into a fixed number of subsets such that a suitable indicator is minimized, taking into account geographical distances and population sizes. Although facility location problems have been the subject of a very large literature, this particular problem in which locations are already fixed but we have to decide which facilities to open in each shift, seems not yet investigated. In a companion paper [2] a special version of this mathematical problem restricted to graphs that are trees has been investigated showing that the problem is polynomial and providing an algorithm for its solution. We show in this paper that the mathematical problem is in general NP-hard. For general questions related to computational complexity the reader is referred for instance to [6].

A problem related to pharmacies is dealt with in [11], where the duty lasts only one day and there is no partition of the pharmacies since the frequency at which a pharmacy must be open depends on the local

population. Furthermore they have to take care that, due to the different Turkish rules, pharmacies can be opened and closed at any time and without any restriction on the sites.

The paper is organized as follows. In Section 2 we provide a mathematical definition of the problem. Then in Section 3 we investigate the computational complexity of some versions of the problem. In Section 4 we present a set packing formulation with exponentially many variables. A branch-and-price approach for this formulation is presented in Section 5, where the pricing subproblem is modeled as a p-median problem with side constraints. In Subsection 5.4 we also present a fast tabu search heuristic to obtain good solutions used as upper bound in the branch-and-price search. A compact Mixed Integer Linear Programming (MILP) model is presented in Section 6. In Section 7 we show a comprehensive set of computational results for real case instances and for random artificial instances. In these tests we compare the two exact methods and the heuristic. Finally we provide some remarks in Section 8.

2 Problem statement

Let F be the set of facilities, i.e., pharmacies and let $n = |F|$. Let C be the set of locations of customers. In each location $i \in C$ there is a population of p_i customers. For each pair $i \in C$, $j \in F$, let d_{ij} be the distance between i and j. For every subset $J \subset F$, let

$$d_i(J) := \min_{j \in J} d_{ij}$$

be the distance of customers in i from the set J.

During a shift some pharmacies must be open. Each pharmacy is open in exactly one shift. Once all pharmacies have completed their duties, the same set of shifts is repeated cyclically. Note that this set of shifts must be a partition of the n pharmacies. Let us call any set of shifts satisfying this partition requirement a *shift partition*. The number of shifts is a fixed number H which is agreed beforehand by all pharmacies. Although not strictly necessary, it is usually required that the shifts are balanced. In this case the number of pharmacies in each shift is either $\lfloor n/H \rfloor$ or $\lceil n/H \rceil$.

We use as an indicator of the quality of a shift partition $J_1, J_2, \ldots, J_H$ the total distance traveled by the customers, assuming that, (i) each customer goes to the closest pharmacy, and (ii) on the average every customer goes to a pharmacy the same number of times in each shift. Assumption (i) is typical in location analysis. Furthermore, due to the type of service pharmacies provide, and taking into account that we are specially interested in the night and holiday service, assumption (i) is typically met in practice. In any case, if a customer prefers going to a distant pharmacy, this extra distance is the choice of the customer and should not be 'charged' to the shift quality. Hence it is natural that an indicator takes into account the distance $d_i(J)$ as defined above. As for assumption (ii), it may be difficult (and beyond the scope of this paper) to know in advance when and if customers will go to pharmacies. Hence we assume that all population behaves in the same way and we consider the simple and equivalent case such that a single customer goes exactly once to a pharmacy in each shift. Observe that, if more specific information on customer behavior is available, it can be used to properly weight the population size p_i. Under these assumptions, the distance associated to a shift J is

$$d(J) = \sum_{i \in C} p_i \, d_i(J)$$

and the total distance traveled by customers, i.e., the cost of the shift partition, may be expressed as

$$\sum_{h \in [H]} d(J_h) \tag{1}$$

(where $[H] := \{1, \ldots, H\}$). Clearly we want to minimize (1) for all feasible shift partitions. From now on we use the more general term *facility* instead of pharmacy.

We observe that every customer has to travel at least the sum of the distances to the H closest facilities, no matter what the shift partition is. Let Δ_i be the sum of the distances from location i to its H closest facilities. We call *utopian optimum* the quantity $\sum_{i \in C} p_i \, \Delta_i$. Every shift partition cost is lower bounded by the utopian optimum.

There are instances for which the optimal solution is the utopian optimum. It has been proved in [2] that if all customer locations and all facilities are the vertices of a tree (or equivalently of a graph with tree metrics) then there exists a partition meeting the utopian optimum and therefore it must be optimal. The tree structure is fundamental for this result. A simple example showing that the utopian optimum and the optimum cost can be different is a circuit with four vertices, three shifts, unit edge lengths and unit populations. Its utopian optimum is equal to 8 (each vertex-customer has as closest vertices-facilities itself, the vertex at right and the vertex at left, for a total distance of 2, which multiplied by four vertices-customers gives 8). However, it is easy to see that in each shift partition two customers have distance 2 but the other two have distance 3 for a total of 10.

We observe also that minimizing an indicator like (1) does not prevent some $d_i(J_h)$ to be too large in the optimal shift partition with respect to some acceptance threshold. Note that, in at least one shift, each customer has to travel to the H-th closest facility whose distance to location i we denote by D_i, i.e., $\max_{h \in [H]} d_i(J_h) \geq D_i$ always hold. Hence by 'too large' we mean a value $d_i(J_h)$ much higher than D_i. In this case we may want to bound $d_i(J_h)$: we will come back to this point later.

3 Computational complexity results

In presenting the computational complexity results we consider the case of unit populations for the sake of simplicity. All results but one will show NP-hardness for the problems, and this clearly implies NP-hardness also for the problems with generic populations. The only polinomiality result remains valid also by extending the problem to generic populations. Let us formally define the problems.

LOCATION PARTITIONING: Given a set F of facilities and a set C of customers, distances $d_{ij} \geq 0$, $i \in C$, $j \in F$, an integer $H \leq n$, a number K, is there a partition of F into subsets $J_1, J_2, \ldots, J_H$, such that $\sum_{h \in [H]} \sum_{i \in C} \min_{j \in J_h} d_{ij} \leq K$?

BALANCED LOCATION PARTITIONING: Given a set F of facilities and a set C of customers, distances $d_{ij} \geq 0$, $i \in C$, $j \in F$, an integer $H \leq n$, a number K, is there a partition of F into subsets $J_1, J_2, \ldots, J_H$, such that $|J_h| \in \{\lfloor n/H \rfloor, \lceil n/H \rceil\}$ for any h, and $\sum_{h \in [H]} \sum_{i \in C} \min_{j \in J_h} d_{ij} \leq K$?

Both problems can be further specialized according to the assumptions we make on distances d_{ij} and on customer and facility sets. We consider the following three versions of the problems in order of decreasing generality:

(a) distances d_{ij} are nonnegative;
(b) customer locations and facilities are two subsets (not necessarily disjoint) of the vertices of a given graph and the d_{ij} are the shortest distances with respect to nonnegative edge lengths of the graph;
(c) as in (b) but the two subsets coincide and include all the vertices of the graph.

Theorem 1. *LOCATION PARTITIONING is NP-complete for $H \geq 3$ and for all three versions.*

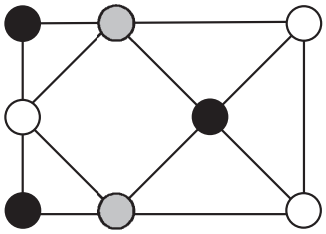

Fig. 1. Domatic number equal to 3

Proof: First note that for version (a) the instance size is $\Omega(n\,|C|)$ due to the list of distances d_{ij} and for versions (b) and (c) is $\Omega(n)$ due to the list of edge lengths. Since the solution size is $O(n\,\log n)$ and computing $\sum_{h\in[H]}\sum_{i\in C}\min_{j\in J_h} d_{ij}$ takes time $O(n\,|C|)$, checking a yes-instance is polynomial with respect to the input length. Hence the problem is in NP.

We prove the NP-completeness via a transformation from DOMATIC NUMBER [4, 6, p. 190]. We recall that, given a graph $G = (V, E)$, DOMATIC NUMBER asks whether there exists a partition of V into at least H dominating subsets. We also recall that a subset of vertices is dominating if each vertex of G is either in the subset or is adjacent to some vertex in the subset. DOMATIC NUMBER is NP-complete for $H \geq 3$.

Given an instance of DOMATIC NUMBER we build an instance of LOCATION PARTITIONING, version (c), as follows (refer to Figure 1): each graph vertex becomes both a customer and a facility and each edge has unit length. The cardinality of the partition is H and the threshold K is set to $K = (H-1)\,|V|$.

Given a feasible instance of DOMATIC NUMBER, we partition the facilities accordingly. For each customer the cost of this partition is $H-1$ and so this is a feasible solution for LOCATION PARTITIONING.

Given a feasible solution for LOCATION PARTITIONING, we note that for each customer the distance is zero for exactly one subset of the partition, whereas for all other subsets the distance is at least one. Hence the cost for each customer is at least $H-1$ and the total cost is at least $(H-1)\,|C| = (H-1)\,|V|$. On the other hand, by assumption, the total cost is at most $(H-1)\,|V|$. Therefore it is exactly equal to $(H-1)\,|V|$. All customers must have the same cost $H-1$, otherwise some customer would have a cost less than $H-1$ and this is impossible. The cost $H-1$ for each customer implies that each partition subset is a dominating set.

Since LOCATION PARTITIONING version (c) is a particular case of the other versions, all three versions are NP-complete. □

The case $H = 2$ presents different complexity results according to the version. As we are going to prove, version (c) is polynomial, whereas the other versions are NP-complete.

Theorem 2. *Versions (a) and (b) of LOCATION PARTITIONING are NP-complete for $H = 2$.*

Proof: We prove the NP-completeness via a transformation from MAX CUT [6, p. 210]. Given a graph $G = (V, E)$ on which we want to solve MAX CUT we build an instance of LOCATION PARTITIONING version (b) on a graph G' obtained from G as follows (refer to Figure 2): each edge $e = (i, j) \in E$ is replaced in G' by the pair of edges (i, k), (k, j) where k is a new vertex for each edge. The set of vertices in G' is the union of the two sets, V' coincident with V and V'' whose elements correspond to the edges in E. Then set $F = V'$, $C = V''$ and assign unit length to each edge in G'. This is version (b) of LOCATION PARTITIONING.

Any bipartition of V corresponds to a bipartition of facilities in G'. If an edge $e \in E$ is in the induced cut in G the corresponding customer in G' has cost 2. If it is not, the cost is at least 4. However, we may freely

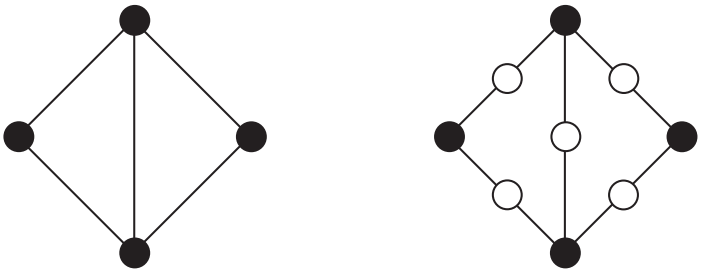

Fig. 2. Transformation from MAXCUT to LOCATION PARTITIONING, $H = 2$

assume that we are dealing with maximal cuts, i.e., no vertex is in the same partition set together with all its neighbors (in this case it could be moved to the other set thereby increasing the cut). This means that the cost of a customer whose corresponding edge is not in the cut is exactly 4.

Hence if there are r edges in the maximal cut the total cost of the corresponding shift partition is $2\,r + 4\,(|E| - r) = 4\,|E| - 2\,r$ and finding a maximum cut is the same as finding a bipartition of minimum cost.

Since LOCATION PARTITIONING version (b) is a particular case of version (a), also version (a) is NP-complete. □

Theorem 3. *LOCATION PARTITIONING, version (c), is polynomial for $H = 2$.*

Proof: For each vertex select the shortest edge incident to that vertex. Break ties according to a fixed arbitrary total order on the edges. Some edges can be selected twice. The selected edges do not form cycles and therefore the vertices can be bicolored with respect to the selected edges, i.e., they can be partitioned into two shifts. Hence a customer has zero length to travel in one shift and the shortest possible distance in the other shift, meeting the utopian optimum. Hence in time $O(|E|)$ we have found the optimum. □

We note that the utopian optimum is reached also if we consider generic population values. Therefore the problem remains polynomial also with generic populations. As far as the balanced case is concerned we have the following result.

Theorem 4. *BALANCED LOCATION PARTITIONING is NP-complete for all three versions if $H \geq 3$, and for versions (a) and (b) if $H = 2$.*

Proof: The problem is in NP by the same arguments as in Theorem 1. We prove the NP-completeness via a transformation from LOCATION PARTITIONING. Given an instance of this problem we build an instance of BALANCED LOCATION PARTITIONING as follows: we create H copies of LOCATION PARTITIONING. Let us denote as F^ℓ and C^ℓ, $\ell = 0, 1, \ldots, H-1$, the copies of facilities and customers respectively, so that in BALANCED LOCATION PARTITIONING the facility set is $\cup_\ell F^\ell$ and the customer set is $\cup_\ell C^\ell$. The distances between customers and facilities within each copy are the given distances d_{ij} of LOCATION PARTITIONING. The distances between customers and facilities of different copies are a large number $D > H\,|C|\,\max_{ij} d_{ij}$. The required number of subsets in the partition is still H and the threshold is $H\,K$. Note that we may assume that for the case $H \geq 3$ we are dealing with version (c) and for the case $H = 2$ we are dealing with version (b). Refer to Figure 3 where version (c) is depicted with $H = 3$.

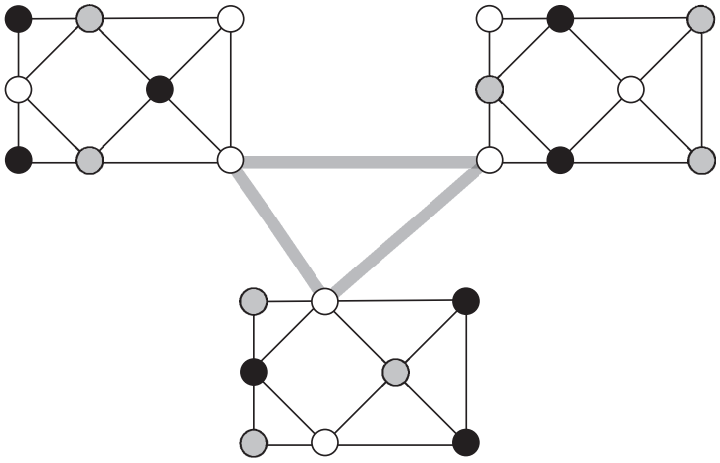

Fig. 3. Transformation from LOCATION PARTITIONING to BALANCED LOCATION PARTITIONING

We now show how, given a feasible instance of LOCATION PARTITIONING, we build a feasible instance of BALANCED LOCATION PARTITIONING. Let us denote the subsets of the partition for LOCATION PARTITIONING as $J_0, J_1, \ldots, J_{H-1}$. Let us denote as $J_h^\ell \subset F^\ell$ the respective copies in BALANCED LOCATION PARTITIONING. Then we partition the subsets in BALANCED LOCATION PARTITIONING as

$$\bigcup_{\ell=0}^{H-1} J^\ell_{(\ell+h) \bmod H} \qquad h = 0, 1, \ldots, H-1\,.$$

In other words we pick up a different subset of the partition from each copy in order to build a partition subset in the larger problem. Hence the cardinality of each partition subset is

$$\sum_{\ell=0}^{H-1} |J^\ell_{(\ell+h) \bmod H}| = \sum_{\ell=0}^{H-1} |J_{(\ell+h) \bmod H}| = |F|\,,$$

so that the partition is balanced. The cost of the instance within each copy is at most K. Hence the cost of the instance is at most $H\,K$.

We now show that given a feasible instance of BALANCED LOCATION PARTITIONING we build a feasible instance of LOCATION PARTITIONING. In any feasible partition of the BALANCED LOCATION PARTITIONING each partition subset must hit a facility in each copy, otherwise the cost of the partition would exceed the threshold $H\,K$ because some customer should go to some facility at distance D. Hence each copy is partitioned in H subsets. Note that each copy can be partitioned in a different way. Let D^ℓ, $\ell = 0, \ldots, H-1$, the cost of each copy given by the induced partitions. The cost of BALANCED LOCATION PARTITIONING is by assumption $\sum_\ell D^\ell \leq H\,K$. Hence in at least one copy ℓ we have a partition with cost D^ℓ at most K. □

We conclude this section by pointing out that LOCATION PARTITIONING, version (c), is polynomial for any H if the underlying graph is a tree as shown in [2] and also if it is a graph with a tree metric.

4 A set packing formulation

Let us denote by $\mathcal{J}$ be the set of all possible shifts, defined either as $\mathcal{J} := 2^F$ if all the facility subsets can be considered (LOCATION PARTITIONING), or as $\mathcal{J} := \mathcal{J}_0$, where $\mathcal{J}_0 = \left\{J \in 2^F : |J| \in \left\{\left\lfloor \frac{n}{H} \right\rfloor \ , \ \left\lceil \frac{n}{H} \right\rceil\right\}\right\}$, in the balanced case.

For any facility subset $J \in \mathcal{J}$, let us define the variable

$$x(J) = \begin{cases} 1 & \text{if } J \text{ is chosen for a shift in a shift partition}\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

and let $a(J)$ be the incidence vector of J, i.e.,

$$a_j(J) = \begin{cases} 1 & \text{if } j \in J\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Hence the (BALANCED) LOCATION PARTITIONING problem can be modeled as

$$\begin{aligned} \min \ & \sum_{J \in \mathcal{J}} d(J)\, x(J)\,, \\ & \sum_{J \in \mathcal{J}} a_j(J)\, x(J) \le 1\,, \quad j \in F\,, \\ & \sum_{J \in \mathcal{J}} x(J) \ge H\,, \\ & x(J) \in \{0,1\}\,, \qquad\qquad J \in \mathcal{J}\,. \end{aligned} \tag{2}$$

Note that, instead of considering the 'natural' set partitioning formulation with constraints

$$\sum_{J \in \mathcal{J}} a_j(J)\, x(J) = 1, \qquad \sum_{J \in \mathcal{J}} x(J) = H,$$

we have chosen a set packing formulation with inequalities. In fact, as we will discuss later, the proposed model has exponentially many variables and can be solved by column generation approaches, and set packing formulations are generally preferred to set partitioning ones [3].

The set packing formulation (2) is valid thanks to the following

Theorem 5. *Given an optimal solution of (2), it is possible to derive an equivalent solution with all inequalities active.*

Proof: Let us first consider the non-balanced case $\mathcal{J} = 2^F$. Adding facilities to a set J does not increase its cost $d(J)$ (and very likely the cost is decreased). Hence, for any solution which does not cover all facilities, we may build another solution covering all facilities with cost not worse than the previous one. If the number of shifts is higher than H, we can iteratively merge two subsets to build a new solution not worse than the original one and with exactly H shifts. Indeed for any two disjoint subsets J' and J'' we have

$$d(J' \cup J'') \le d(J') \quad \wedge \quad d(J' \cup J'') \le d(J'') \quad \Longrightarrow$$

$$d(J' \cup J'') \le 2\, d(J' \cup J'') \le d(J') + d(J'')\,.$$

It follows that, starting from the given optimal solution of (2), we can obtain another solution, with all constraints active, which is also optimal since it is not worse than the given optimal one.

As for the balanced case $\mathcal{J} = \mathcal{J}_0$, we consider two cases: either n is multiple of H or not.

If n is multiple of H, by a simple counting argument any feasible shift partition must have H shifts and cover all facilities, so that all the constraints in the given optimal solution are active.

If n is not a multiple of H then let H_0 be the number of subsets of cardinality $\lfloor n/H \rfloor$ and let H_1 be the number of subsets of cardinality $\lceil n/H \rceil$ in the given optimal solution. Assume that there are k extra sets, i.e.,

$$H_0 + H_1 = H + k\,. \tag{3}$$

Note that $H_0 \geq k$ (otherwise, from (3), $H_1 > H$ and this is impossible), and, in order to obtain a new solution with exactly H shifts, we may delete k sets of cardinality $\lfloor n/H \rfloor$ and put their elements one by one into the remaining $H_0 - k$ sets, so that these sets become of cardinality $\lceil n/H \rceil$. For this to be possible we need to show that H_0 is sufficiently large and, more precisely, that

$$H_0 - k \geq k \left\lfloor \frac{n}{H} \right\rfloor . \tag{4}$$

To this end, let $r = \lceil n/H \rceil - n/H = \lfloor n/H \rfloor + 1 - n/H$. From the packing constraints (first set of inequalities in (2)) we have

$$H_0 \left\lfloor \frac{n}{H} \right\rfloor + H_1 \left\lceil \frac{n}{H} \right\rceil = H_0 \,(\frac{n}{H} + r - 1) + H_1 \,(\frac{n}{H} + r) = (H + k)\,(\frac{n}{H} + r) - H_0 \leq n$$

which implies

$$H_0 \geq k\,(\frac{n}{H} + r) + H\,r \geq k\,(\frac{n}{H} + r)$$

and

$$H_0 - k \geq k\,(\frac{n}{H} + r - 1) = k \left\lfloor \frac{n}{H} \right\rfloor ,$$

which is the needed relation (4). Hence building the new balanced solution with exactly H shifts is possible. Now, the cost of the new solution can be obtained from the given one by subtracting the non-negative costs of the r deleted shifts and by substituting the costs of the r shifts with augmented cardinality with a new cost. As each of these shifts is obtained by adding one facility, the new cost is not worse than the previous one, so that the overall cost of the new solution is not worse than the given one. Finally, if not all facilities are covered, we may add the non covered facilities to the sets of smaller cardinality. This is clearly possible without violating the balancing constraint since $n < H\,\lceil n/H \rceil$, and allow us obtaining another balanced solution, with all constraints active, which is also optimal since it is not worse than the given optimal one.

□

The proposed model (2) has an exponential number of variables and a branch-and-price approach based on solving its linear relaxation by column generation has been developed, as described in the next section.

5 Solving the set packing model by branch-and-price

Model (2) is solved by a branch-and-price procedure using the components described in the following.

5.1 Initial set of variables

The initial set of variables is given by a greedy procedure that starts from empty shifts and iteratively adds a facility to a shift, until all the facilities are assigned. At each iteration, the facility and the shift are chosen

as follows: for each pair of not-yet-assigned facility j and shift k, we compute the decrement of the cost of k obtained by adding j to k, and we select the pair associated to the largest decrement. In order to obtain feasible solutions for the balanced case, the number of facilities in each initial shift is determined a-priori, and shifts are not considered once they are completed.

5.2 Pricing subproblem

Noting that the binary requirement for variables $x(J)$ can be relaxed as $x(J) \geq 0$, the dual of the continuous relaxation of (2) is

$$\begin{aligned} \max \; & -\sum_{j \in F} v_j + H\, u\,, \\ & -\sum_{j \in F} a_j(J)\, v_j + u \leq d(J)\,, \qquad J \in \mathcal{J}\,, \\ & u \geq 0, v_j \geq 0\,, \qquad\qquad\qquad\quad j \in F\,. \end{aligned} \tag{5}$$

Hence the pricing problem at the root node consists in solving

$$w = \min_{J \in \mathcal{J}} \; \Big\{ d(J) + \sum_{j \in J} v_j \Big\} \tag{6}$$

and checking whether $w \geq u$. If so, we have reached optimality of the relaxation, otherwise we must introduce a set $\hat{J}$ corresponding to the minimum in (6) into the master problem, i.e., problem (2) restricted to a subset of shifts, and iterate.

Problem (6) to be solved at the root node is the p-median problem [8–10]. Unfortunately the p-median problem is NP-hard ([6] p. 220), but the classical MILP model performs reasonably well and there is available a large number of good heuristics. The fact that the p-median problem is polynomial for *fixed* p (i.e., if p does not enter the instance definition), which is the case for balanced shifts, does not help very much for computational purposes [12]. It remains a hard problem. Further, at nodes different from the root one, side constraints related to branching decisions have to be considered.

Since actually we don't need the exact optimum of (6) but we may carry out the computation with any $\hat{J}$ such that $u > d(\hat{J}) + \sum_{j \in F} a_j(\hat{J})\, y_j$, it is computationally convenient to price with a fast heuristic and, only in case the heuristic fails, we resort to an exact pricing procedure.

For heuristic pricing we have developed a simple local search procedure. It starts from a random (in case balanced) shift J chosen as the current solution, and it obtains perturbed neighbor solutions J' by *adding* a facility $k \notin J$ ($J' := J \cup \{k\}$), *removing* a facility $h \in J$ ($J' := J \setminus \{h\}$), and *swapping* two facilities $k \notin J$ and $h \in J$ ($J' := J \setminus \{h\} \cup \{k\}$). In the balanced case, just the perturbations yielding to shifts with feasible cardinality are considered. Neighbor solutions are evaluated according to two criteria in a lexicographic fashion: the number of violated branching constraints (as we will discuss later) and the value of the pricing function in (6). If the best neighbor solution J' is better than the current one, J' becomes the new current solution and the algorithm iterates, otherwise the algorithm stops returning the local minimum J'.

The exact pricing procedure is based on the following classical MILP model where the variable $t_j \in \{0, 1\}$ denotes if facility j is chosen and variable s_{ij} denotes if the customer i goes to facility j. Due to the objective function each customer always goes to the closest open facility. For the (BALANCED) LOCATION

PARTITIONING the pricing problem at the root node is

$$\begin{aligned}
\min \quad & \sum_{j\in F}(v_j\, t_j + \sum_{i\in C} p_i\, d_{ij}\, s_{ij})\,, & \\
& \sum_{j\in F} s_{ij} = 1\,, & i \in C\,, \\
& \big(\ \lfloor n/H \rfloor \le \sum_{j\in F} t_j \le \lceil n/H \rceil\ , & h \in [H]\ \ ,\big) \\
& s_{ij} \le t_j\,, & j \in F,\ i \in C\,, \\
& t_j \in \{0,1\}\,,\ s_{ij} \ge 0\,, & j \in F,\ i \in C\,.
\end{aligned} \tag{7}$$

Note that the binary variables s_{ij} have been relaxed. As already remarked, the customers in each location go to the closest open facility and, if the optimal solution of (7) has fractional s_{ij} values, this means that more than one facility are at the minimum cost for customers in location i. However, if this is the case, no fractional solution can be a vertex solution and the simplex method will output just a binary solution.

Branching requirements are handled by adding further constraints, as we will discuss later. Similarly, the pricing problem allows us to easily control the generation of sets J in (7) if additional constraints are needed. For example, as observed before, we may be interested in generating shifts inducing limited travel lengths, and therefore we can impose a threshold $\overline{D}_i$ on the distances traveled by customers in location i. This can be realized by adding the constraint

$$\sum_{j\in F} s_{ij}\, d_{ij} \le \overline{D}_i\,, \qquad i \in C\,,$$

to (7), and the property that s_{ij} need not to be imposed binary a priori is still preserved.

5.3 Branching strategy

Thanks to Theorem 5, an optimal solution to (2) exists corresponding to a set partition and we can apply the Ryan and Foster branching rule [13], which performs better than branching on variables x and allows us to consider the branching choices in the pricing subproblems without additional constraints on the master problem.

The rule is based on the fact that a solution to (2) is integral if and only if, for each pair of constraints related to facilities i and j, the sum

$$\tilde{X}_{ij} = \sum_{J\in\mathcal{J}: i\in J, j\in J} x(J)$$

of the values of the variables x contained in both constraints is integral, i.e., either 1, if and only if the two facilities are "packed" in the same shift, or 0 if and only if the two facilities are "packed" in different shifts. Hence, given a fractional solution, we select two constraints such that $0 < \tilde{X}_{ij} < 1$ (in our implementation, i and j are selected such that $\tilde{X}_{ij}$ is as close to 0.5 as possible). Then we create two branches by forcing $\tilde{X}_{ij}$ to, respectively, 1 ("same" branch) or 0 ("different" branch), which can be done directly in the pricing problem (6), by imposing the additional constraints, respectively, $t_i = t_j$ or $t_i + t_j \le 1$. As a consequence, the slave problem to be solved at each node of the branch-and-bound tree is (7) with a set of additional branching constraints related to the node itself and its predecessors. Again, these additional constraints preserve the property that s_{ij} need not to be imposed binary a priori.

The same constraints should be satisfied by the solution provided by the local search heuristic for pricing. To this end, as we have mentioned above, instead of forbidding non-feasible solutions (which may lead to poor neighborhoods), we penalize them with the number of violated branching constraints: the better is a neighbor solution, the less branching constraints it violates or, the number of violations being equal, the smallest is the value of the objective function in (6). Of course, if the returned local optimum is not a feasible shift, heuristic pricing fails and we resort to the exact procedure.

5.4 Upper Bound procedure

As we will see in the next section devoted to computational results, the relaxation of model (2) provides a tight lower bound for the (BALANCED) LOCATION PARTITIONING problem. Further speed up may be obtained by quickly generating a good initial incumbent solution and, hence, a good initial upper bound. To this end, we propose a straightforward implementation of a tabu search [7] heuristic.

The initial solution is the shift partition given by the greedy procedure described in Subsection 5.1. Given a shift partition, represented with the subsets of facilities corresponding to shifts, neighbor solutions are obtained by *transferring* a facility to an alternative shift, or by *swapping* facilities belonging to different shifts. Starting from the initial solution, all possible neighbors are generated and evaluated. Then, the algorithm moves to the best neighbor, in case updating the value of the best available solution, and iterates. The search stops after K (parameter to be calibrated) consecutive iterations that do not improve over the best solution. At each iteration, the best neighbor may be worst than the current solution, leading to possible loops (the algorithm may come back to an already visited solution and indefinitely cycle). To avoid cycling, tabu search excludes from the neighborhood the solutions related to more recent moves [**?**, see]]gloverts: in our case, we exclude the solutions obtained by transferring or swapping facilities involved in the last T (parameter to be calibrated) moves, unless they are better than the best available solution.

6 A MILP compact formulation

We have also taken into account a compact formulation for the (BALANCED) LOCATION PARTITIONING problem, with binary variables z_{jh} to denote whether facility j is open on shift h or not, and binary variables y_{ijh} to denote whether customer i goes to facility j on shift h or not. The MILP model is

$$
\begin{aligned}
\min \quad & \sum_{h\in[H]}\sum_{i\in C}\sum_{j\in F} p_i\, d_{ij}\, y_{ijh}\,, & & \\
& \sum_{h\in[H]} z_{jh} \;= 1\,, & & j\in F\,,\\
& \sum_{j\in F} y_{ijh} = 1\,, & & i\in C, h\in[H]\,,\\
& \quad y_{ijh} \le z_{jh}\,, & & i\in C, j\in F, h\in[H]\,,\\
& \big(\ \lfloor n/H\rfloor \le \sum_{j\in F} z_{jh} \le \lceil n/H\rceil\ , & & h\in[H]\,,\ \big)\\
& z_{jh}\in\{0,1\}\,, y_{ijh}\ge 0\,, & & i\in C,\ j\in F,\ h\in H\,.
\end{aligned}
\tag{8}
$$

Note that the binary variables y_{ijh} have been relaxed. By the objective function, the customers in each location go to the closest open facility and, as for model (7), the simplex method will output a binary solution.

The advantage of this compact model with respect to the column generation model of the previous section relies on the fact that it can be simply fed to any MILP solver without additional implementation effort. On the other hand the lower bound provided by the integrality relaxation of (8) is much poorer than the one given by the relaxation of (2), therefore leading to a larger branch-and-bound tree.

Indeed, it turns out that the lower bound of the relaxation of (8) is equal to the utopian optimum. For each customer $i \in C$ let J_i be the set of the H closest facilities to i. Put $z_{jh} = 1/H$ for all j and h and $y_{ijh} = 1/H$ if $j \in J_i$, 0 otherwise. Then we have $\sum_{h\in[H]} z_{jh} = 1$ and $\sum_{j\in F} y_{ijh} = \sum_{j\in J_i} 1/H = 1$. The other constraints are trivially satisfied and this solution is feasible for the relaxation of (8). Moreover the objective function value is

$$\sum_{h\in[H]} \sum_{i\in C} \sum_{j\in F} p_i\, d_{ij}\, y_{ijh} = \sum_{h\in[H]} \sum_{i\in C} \sum_{j\in J_i} p_i\, d_{ij}\, \frac{1}{H} =$$

$$\sum_{i\in C} p_i \sum_{j\in J_i} \sum_{h\in[H]} d_{ij}\, \frac{1}{H} = \sum_{i\in C} p_i \sum_{j\in J_i} d_{ij} = \sum_{i\in C} p_i\, \Delta_i\,,$$

i.e., the utopian optimum. This is also a lower bound for any feasible fractional solution. Indeed, if we denote $w_{ij} = \sum_h y_{ijh}$, the constraints of (8) imply $w_{ij} \le 1$ and $\sum_j w_{ij} = H$. The objective function can be rewritten as $\sum_{i\in C} p_i \sum_{j\in F} d_{ij}\, w_{ij}$, which is minimized by assigning, for each i, the value 1 to variables w_{ij} corresponding to the H minimum values d_{ij}.

As we will see in the section devoted to computational results, although solving model (8) with state-of-the-art solvers is competitive with (2) on small instances, it becomes prohibitively slow on medium size instances, due to the worse lower bound of the relaxation, the high number of variables and the high degree of symmetry.

In order to decrease the degree of symmetry we consider a subset $\{j_1, j_2 \dots j_H\}$ of facilities and add to (8) the following symmetry breaking constraints:

$$\sum_{h=1}^{r} z_{j_r h} = 1\,, \qquad r \in [H]\,, \tag{9}$$

meaning that pharmacy j_1 has to be included in shift 1, j_2 in shift 1 or 2, j_3 in shift 1, 2 or 3 and so on. The constraints are more effective (i.e. there is no other solution obtained by simply permuting the shifts) if facilities $j_1, \dots, j_H$ are in different shifts. Heuristically we may think that this situation is more likely to happen if the facilities in (9) are chosen as: j_1 is the facility that minimizes the total weighted distance from every customer, j_2 the second best facility and so on until facility j_H.

7 Computational results

In the previous sections we have described three ways of computing a solution for the (BALANCED) LOCATION PARTITIONING problem. Two of them are exact and are based on MILP formulations. The first one is the branch-and-price procedure described in Sections 4 and 5 (referred to as "B&P") and the second one is the compact MILP model (8) using the symmetry-breaking constraints (9) as described in Section 6 (referred to as "Compact") . The third one is heuristic and is the tabu search procedure ("TS") described in Subsection 5.4. These three methods have been tested on all instances described next thus providing a comparison among the methods.

The branch-and-price algorithm has been implemented in C++, using the Scip 2.1.1 framework [1] and Cplex 12.6 as linear programming engine. The compact model (8) with constraints (9) was directly solved by Cplex 12.6 as MILP solver using default settings. Both branch-and-price and Cplex use the value of the initial incumbent solution given by the upper bound procedure described in Subsection 5.4. Results in the following tables refer to a single thread on Intel Dual Core 2.7 GHz CPU with 4 GB RAM. According to preliminary calibration, tabu search parameters have been set to $K = \min\{10 \cdot |F|, 500\}$ and $T = \min\{|F|/3, 21\}$, where $|F|$ is the number of facilities.

We have two classes of instances: the first class is based on real data drawn from local Pharmacy Associations in Italy, the second class is made of randomly generated instances. All instances can be found at [5], with all necessary instructions.

Concerning the first class of instances, two instances in this set refer to two sub-areas of the Padova province (instances PD1 and PD2) and two other instances refer to two sub-areas of the Friuli-Venezia Giulia region (instances FVG1 and FVG2). Both the Padova province and the Friuli-Venezia Giulia region are located in North-Eastern Italy.

Instance PD1 has 55 customer locations and 55 pharmacies (one per location), PD2 has 49 locations and 49 pharmacies (again one per location), FVG1 has 20 locations and 23 pharmacies (one per location but two locations with two and three pharmacies respectively) and FVG2 has 97 locations and 97 pharmacies, one per location. Different population sizes are usually not taken into account by the local Pharmacy associations and therefore we have run these instances by taking $p_i = 1$, for all $i \in C$. However, we did have available the population sizes for the two instances of the Padova province and so we have run these instances also by considering the actual populations.

According to Pharmacy association requirements, the instances PD1 and PD2 have 11 shifts and the instances FVG1 and FVG2 have 9 shifts. However, for the sake of completeness of the computational tests, we have considered the cases with 3, 6, 9 and 11 shifts (except for FVG2), both balanced and unbalanced.

In Tables 1, 2, 3 and 4 we report the computational results for the first class of instances. In these tables each row refers to one instance only. For each solution method we report the computing time in seconds indicated by "t" and the gap indicated by "g". In more detail the gap for the B&P and the compact method is the integrality gap (measured as a percentage), i.e., the difference between the optimal value and the optimal value of the integrality relaxation divided by the former value and multiplied by 100, whereas the gap for the tabu search is its relative error, i.e., the difference between the tabu search value and the optimal value divided by the optimal value and multiplied by 100. When the gap is exactly zero we write the figure "0" in the tables. When the gap is so small that its two digit approximation is zero we write instead "0.00".

We have set a time limit of four hours for this first class of instances. Table 1 refers to the unbalanced case with unit populations, Table 2 to the balanced case with unit populations, Table 3 to the unbalanced case with actual populations and Table 3 to the balanced case with actual populations. Note that the branch-and-price procedure uses random components (the initial solution of the pricing heuristic): it has been run 10 times and average results are shown. As already remarked, we have run only the Padova province instances for the case of actual populations.

Some instances, marked with (*) and (**) in the tables, could not be solved within the time limit. For the instances marked with (*) the gap has been computed by using the known optimal value obtained by the other method. For the instances marked with (**) the optimal value is not available and the gap has been computed by using the best known dual value.

B&P has been able to solve all the proposed instances within the time limit (and using fairly less than half an hour in almost all cases), with the only exception of the large FVG2 instance (which could not be solved by Cplex neither). This is mostly due to the tightness of the set packing formulation (2): the value of its

integrality relaxation is often equal to the optimal integer value, and, in the remaining cases, it is very close to it. Beyond the large FVG2 instance, Cplex fails in solving five additional cases related to PD1 instance with 11 and 9 shifts. This is to be ascribed to a poorer integrality relaxation of the compact formulation (the gap is up to 0.92% and just in a few cases below 0.1%) as well as to the model size. Concerning computational times, Cplex is normally faster for the small FVG1 instance and when 3 shifts are considered, whereas B&P works better with larger numbers of shifts. We notice also that B&P running times are sensible to population size, with doubled figures (on average) for the case of actual population size.

The second class of instances are random instances generated as follows. We have separately considered versions (a), (b) and (c) of the problem (refer to Section 3). For each instance we have generated a certain number of independent uniformly distributed random points in a unit square. The distance d_{ij} between any two points is an integer number obtained by rounding the Euclidean distance between the two points multiplied by 100. For each instance of version (a) we have generated a set C of points as customer locations and a set F of points as facilities. For each instance of version (b) we have generated a set C of points as customer locations and, among these points, we have randomly selected a subset F of points as facility locations. For each instance of version (c) we have generated a set F of points, each one being both a location and a facility. Hence we have implicitly assumed that the underlying graphs for versions (b) and (c) are complete graphs with arc lengths given by Euclidean distances.

We have chosen 12 different instance types by varying $(|C|, |F|, H)$ for versions (a) and (b) as: $(20, 10, 5)$, $(40, 20, 5)$, $(40, 20, 10)$, $(60, 30, 5)$, $(60, 30, 10)$, $(60, 30, 20)$, $(80, 40, 5)$, $(80, 40, 10)$, $(80, 40, 20)$, $(100, 50, 5)$, $(100, 50, 10)$, (100, 50,20), and 14 different instance types for version (c) as $(20, 20, 5)$, $(20, 20, 10)$, $(40, 40, 5)$, $(40, 40, 10)$, $(40, 40, 20)$, $(60, 60, 5)$, $(60, 60, 10)$, $(60, 60, 20)$, $(80, 80, 5)$, $(80, 80, 10)$, $(80, 80, 20)$, $(100, 100, 5)$, $(100, 100, 10)$, (100, 100,20). For each instance type we have generated 10 random instances.

The 10 random instances are stochastically independent. However, the k-th instance of each group of 10 is related to the k-th instance of all other groups, because the random points in the unit square have been generated with the same seed. Hence, e.g., the 20 points of the k-th instance of the group $(20, 20, h')$ are a subset of the 40 points of the k-th instance of the group $(40, 40, h'')$ (for any h', h''), and similarly the 40 points of the k-th instance of the group $(40, 40, h')$ are a subset of the 60 points of the k-th instance of the group $(60, 60, h'')$, and so on.

All instances have been computed both for the unbalanced case and for the balanced case and also both with unit populations and with varying populations. In the latter case the populations are uniformly distributed random numbers between 10 and 100. As before the populations of the 10 instances are stochastically independent and the k-th instances of each group have the same populations for the common locations.

Due to the large amount of computation required for all these parameter combinations, we have reduced the time limit to 1800 seconds.

The parameters defining the instances and related computational results are reported in Tables 5, 6, 7 and 8. In the tables for the second class of instances each row refers to the ten instances of each group. On each row we report, as before, two output data, i.e., the computing time in seconds (t) and the gap (g) as a percentage. However, differently from the real data, these value are averages over the ten instances. In case not all ten instances could be solved within the time limit the average is computed only with respect to the solved instances. Moreover, we indicate also the number of instances which could be solved within the time limit of 1800 seconds in the column labeled with "s" (for "solved"). Clearly this is unnecessary for the tabu search method. On some entries of these columns the number of solved instances is preceded by a number in a smaller font and within parenthesis. This number refers to the number of instances whose integrality relaxation could be solved within the time limit only when this number is smaller than ten. In the vast

majority of cases the integrality relaxations of all ten instances could be solved within the time limit and therefore in these cases the number '10' is not reported.

Table 5 refers to the unbalanced case with unit populations, Table 6 to the balanced case with unit populations, Table 7 to the unbalanced case with varying populations and Table 8 to the balanced case with varying populations. Again, some instances are marked with (*) and (**), with the same meaning as before. Moreover, in some cases, marked with "n.a." (not available), no relaxation could be solved within the time limit and the related gap could not be computed.

The results confirm the trend observed for the first class of instances. A general pattern which emerges from the computations is that the compact MILP model performs better than the B&P algorithm for small instances, whereas the opposite happens with large instances. The number of shifts plays also an important role. Generally, by increasing the number of shifts the compact model tends to have prohibitive running times and B&P is the only option for an exact solution. This is particularly evident for the quite large instances with 80 and 100 locations for both facilities and customers, where none of the 10 instances with few shifts could be solved by the B&P whereas the other instances with more shifts could not be solved by the compact model.

As expected form the theory, the integrality relaxation gap for the column generation model is almost negligible and for the small and medium size instances is just zero most of the times. In spite of this fact, the running times remain high, at least for the medium and large size instances. Looking at the time spent for solving the LP at the root node of the branch-and-bound tree (this data is not reported on the tables) we have observed that in almost all instances, whenever the gap is exactly zero, then roughly 99% of the computing time is spent at the root node. This means that the branching strategy is effective in finding the integral solution. Apparently, the column generation procedure is time consuming and this is no surprise since it relies on an NP-hard problem and may also require a very large number of columns (which are not always found by the heuristic) before reaching optimality. When the gap is not zero, though small, the time spent in searching the branch-and-bound tree is comparable to the time spent in solving the root node.

As already observed for real instances, the impact of the population sizes has dramatic effect on the running times of the B&P model. On the average the running times increase between one and two times, whereas the running times of the compact model decrease. However, the pattern according to which B&P is faster for larger instances with larger number of shifts is preserved. The tabu search times are almost unaffected by the populations sizes.

As for the question of balanced vs unbalanced instances the running times for the unbalanced B&P are slightly worse than the balanced cases. This is an expected result because the number of possible subsets to be generated is much higher for the unbalanced case.

We also note that the tabu search procedure is very fast in determining very good solutions. In almost all instances the error was less than 1% and, normally, between 0 and 0.3%, which, for practical purposes may be totally acceptable, and the solution is found quite quickly, at least in comparison to the exact methods (order of seconds for medium-size instances and a few minutes for the large ones). It is also true that, for the kind of problem we are considering, i.e., finding a solution that is eventually implemented to last for a long time, computations taking several hours can also be considered acceptable and the pursue of a 'very good' and 'certified' solution makes more sense.

8 Conclusions

Starting from real cases related to pharmacy opening-shifts planning, we have addressed the problem of determining an optimal distribution of facilities (pharmacies) into a given number of subsets with the aim of

minimizing the overall distance traveled by customers to reach the closest facility in each subset. The problem merges the requirements of partitioning problems (facilities have to be partitioned into shifts) and location problems (shifts should minimize distances) and, to the best of our knowledge, it is new to Operations Research literature. We have called it LOCATION PARTITIONING problem and we have studied its computational complexity, taking also into account a possible variant of practical relevance called BALANCED LOCATION PARTITIONING, where the difference between shift cardinalities is at most 1.

For both variants, we have considered three sub-cases depending on how the distances between customers and facilities (pharmacies) are defined, as we briefly recall here: (a) distances are simply non-negative, (b) customers and facilities are located on a network and distances are the length of the shortest paths, (c) as in (b), with both customers and facilities located in every nodes. Starting from the NP-completeness of the DOMATIC NUMBER problem, we have shown that for both the balanced and the non-balanced variants, the problem is NP-Hard if the number of shifts $H \geq 3$, independently from distance definition. For the case $H = 2$, complexity may depend on both the variant and the distance definition: the problem is always NP-Hard in case (a) and (b), while the non-balanced case (c) is polynomial.

In order to solve some real cases issued by local Pharmacists' associations in Italy, we have proposed two alternative MILP models: a set packing formulation with an exponential number of binary variables associated to possible shifts, and a compact formulation with binary variables related to the assignment of facilities to shifts. For the first formulation, we have devised a branch-and-price algorithm using, for pricing, a local search heuristic and an exact formulation as p-median problem with side constraints.

The second model has been directly solved by Cplex. The two formulations, as well as a tabu search heuristic, have been also tested on a set of randomly generated instances, involving both balanced and unbalanced variants, the three problem sub-cases, and unit or varying population size. The proposed branch-and-price procedure solves to proven optimality all the real cases under study but one large instance with 97 locations and facilities, thanks to the good quality of the integrality relaxation, whose value is very close (and often equal) to the optimal integer solution. Running times are longer than Cplex on the compact formulation if the number of shift is small but remain stable or even decrease with more shifts, where Cplex running times tend to become prohibitively large. Concerning the tabu search heuristic, it can be used to quite quickly obtain near optimal solutions and to tackle large instances.

The results show that LOCATION PARTITIONING problems, even if they are NP-Hard, can be solved in cases of practical relevance, at least for medium-size instances. Also, there is room for reducing running times and increasing the size of solvable instances by, for example, improving the pricing heuristic or by stabilizing the column generation, which is beyond the scope of the paper and may be the object of further research. From a theoretical point of view, the study of case (c) of the BALANCED LOCATION PARTITIONING problem with $H = 2$ would complete the discussion on computational complexity, and it is the object of ongoing investigation.

References

1. T. Achterberg. 2009. SCIP: Solving constraint integer programs. *Mathematical Programming Computation* **1:1** 1–41.
2. G. Andreatta, L. De Giovanni and P. Serafini. 2014. Optimal Shift Coloring of Trees. *Operations Research Letters* **42** 251–256. DOI: 10.1016/j.orl.2014.04.004.
3. C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsberg and P.H. Vance. 1998. Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research* **46** 316–329.
4. E.J. Cockayne and S.T. Hedetniemi. 1975. Optimal domination in graphs. *IEEE Trans. Circuits and Systems* **22** 855–857.

5. L. De Giovanni. 2014. Location partitioning instances. Retrieved: July 25th, 2014, `http://www.math.-unipd.it/∼luigi/LPP/LocationPar titioningInstances.zip`.
6. M. R. Garey and D. S. Johnson. 1979. *Computers and Intractability: A guide to the theory of NP-completeness*. W. H. Freeman and Co., San Francisco.
7. F. Glover and M. Laguna. 1997. *Tabu search*. Kluwer Academic, Boston.
8. S. L. Hakimi. 1964. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research* **12** 450–459.
9. S. L. Hakimi. 1965. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research* **13** 462–475.
10. P. B. Mirchandani and R. Francis, editors. 1990. *Discrete location theory*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., New York.
11. Ö. Özpeynirci and E. Ağlamaz. 2012. Mathematical Models for Pharmacy Duty Scheduling, in: *Proceedings of ECCO 2012 - 25th Conference of European Chapter on Combinatorial Optimization*, April 26-28, 2012, Antalya, Turkey.
12. J. Reese. 2006. Solution methods for the p-median problem: An annotated bibliography. *Networks* **48(3)** 125–142.
13. D. Ryan and B.A. Foster. 1981. An integer programming approach to scheduling, in: *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, A. Wren, ed., North Holland, Amsterdam, p. 269–280.

Instance type				B&P		Compact		TS	
Name	$\|C\|$	$\|F\|$	H	t	g	t	g	t	g
FVG1	20	23	3	0.12	0	0.08	0.24	0.18	0
FVG1	20	23	6	1.25	0	0.25	0.15	0.39	0
FVG1	20	23	9	0.54	0	0.17	0	0.53	0.35
FVG1	20	23	11	0.32	0	0.22	0	0.83	0
PD1	49	49	3	11.71	0	0.36	0.54	1.28	0
PD1	49	49	6	86.15	0	731.07	0.64	13.86	0.26
PD1	49	49	9	79.01	0	3345.76	0.65	14.69	0.18
PD1	49	49	11	229.91	0.03	(*)	0.85	16.42	0.19
PD2	55	55	3	1227.17	0	0.71	0.11	2.18	0
PD2	55	55	6	717.06	0	878.43	0.32	7.02	0.18
PD2	55	55	9	589.47	0.01	1316.88	0.22	15.70	0.12
PD2	55	55	11	81.82	0	938.84	0.16	40.33	0.06
FVG2	97	97	9	(**)	0.01	(**)	0.50	191.30	0.47

Table 1. First class of instances - unbalanced case - unit populations

Instance type				B&P		Compact		TS	
Name	$\|C\|$	$\|F\|$	H	t	g	t	g	t	g
FVG1	20	23	3	0.10	0	0.11	0.24	0.18	0
FVG1	20	23	6	0.52	0	0.48	0.33	0.41	0
FVG1	20	23	9	0.59	0	0.19	0	0.34	0.25
FVG1	20	23	11	0.47	0	0.26	0	0.41	0
PD1	49	49	3	6.32	0	0.56	0.54	2.26	0
PD1	49	49	6	166.16	0	544.13	0.74	3.60	0.36
PD1	49	49	9	147.60	0.01	3772.67	0.64	7.67	0.13
PD1	49	49	11	201.86	0.02	(*)	0.83	19.48	0.19
PD2	55	55	3	466.17	0	7.93	0.11	2.03	0
PD2	55	55	6	375.83	0	1764.58	0.32	15.64	0.07
PD2	55	55	9	525.92	0.02	2988.89	0.24	26.59	0.09
PD2	55	55	11	94.27	0	879.89	0.18	32.64	0.08
FVG2	97	97	9	(**)	0.01	(**)	0.52	127.95	0.44

Table 2. First class of instances - balanced case - unit populations

Instance type				B&P		Compact		TS	
Name	$\|C\|$	$\|F\|$	H	t	g	t	g	t	g
PD1	49	49	3	918.19	0	1.11	0.59	1.60	0.23
PD1	49	49	6	213.33	0	471.10	0.64	6.78	0.34
PD1	49	49	9	492.60	0.00	(*)	0.65	25.18	0.11
PD1	49	49	11	485.51	0.02	(*)	0.92	19.62	0.19
PD2	55	55	3	1231.28	0	0.67	0.03	2.56	0.03
PD2	55	55	6	907.04	0	903.42	0.26	13.77	0.09
PD2	55	55	9	233.53	0	602.61	0.14	20.73	0.19
PD2	55	55	11	174.01	0	556.11	0.11	63.00	0.03

Table 3. First class of instances - unbalanced case - actual population sizes

Instance type				B&P		Compact		TS	
Name	$\|C\|$	$\|F\|$	H	t	g	t	g	t	g
PD1	49	49	3	11.57	0.00	1.37	0.59	1.59	0
PD1	49	49	6	225.12	0	1018.55	0.70	11.25	0.13
PD1	49	49	9	264.35	0.01	13747.77	0.64	13.92	0.18
PD1	49	49	11	344.51	0.02	(*)	0.88	17.11	0.20
PD2	55	55	3	1170.04	0	0.63	0.03	2.28	0.03
PD2	55	55	6	1969.45	0.01	4637.51	0.27	9.45	0.15
PD2	55	55	9	1133.06	0.01	1688.85	0.18	28.14	0.19
PD2	55	55	11	107.48	0	540.23	0.13	38.51	0.05

Table 4. First class of instances - balanced case - actual population sizes

Instance type				B&P			Compact			TS	
version	$\|C\|$	$\|F\|$	H	s	t	g	s	t	g	t	g
(a)	20	10	5	10	0.11	0	10	0.04	0.01	0.08	0
(a)	40	20	5	10	4.10	0	10	0.53	0.09	0.87	0.04
(a)	40	20	10	10	0.70	0.01	10	1.43	0.12	1.10	0.06
(a)	60	30	5	10	19.55	0	10	2.08	0.09	1.28	0.05
(a)	60	30	10	10	9.47	0	10	15.73	0.08	5.17	0.05
(a)	60	30	20	10	2.74	0	10	200.77	0.26	21.43	0.15
(a)	80	40	5	10	97.39	0	10	30.01	0.23	4.53	0.16
(a)	80	40	10	10	51.39	0	9	140.52	0.14	25.99	0.11
(a)	80	40	20	10	13.31	0	8	554.81	0.18	65.21	0.05
(a)	100	50	5	10	592.41	0.01	10	316.48	0.27	11.41	0.18
(a)	100	50	10	10	302.70	0	4	927.24	0.18	54.46	0.14
(a)	100	50	20	10	72.07	0	1	1416.46	0.27	176.84	0.04
(b)	20	10	5	10	0.14	0	10	0.03	0	0.08	0
(b)	40	20	5	10	2.12	0	10	0.58	0.08	0.56	0.07
(b)	40	20	10	10	0.88	0.01	10	2.11	0.12	1.09	0.03
(b)	60	30	5	10	14.58	0	10	6.83	0.18	1.62	0.08
(b)	60	30	10	10	8.19	0	10	38.89	0.07	5.45	0.11
(b)	60	30	20	10	2.56	0	10	162.73	0.27	21.27	0.03
(b)	80	40	5	10	80.48	0	10	52.09	0.27	4.34	0.13
(b)	80	40	10	10	62.38	0	9	370.39	0.18	22.28	0.08
(b)	80	40	20	10	15.19	0	7	847.86	0.15	60.67	0.08
(b)	100	50	5	10	343.12	0	10	166.81	0.26	13.62	0.20
(b)	100	50	10	10	178.53	0	1	71.97	0.20	57.05	0.10
(b)	100	50	20	10	52.48	0	0	(*)	0.28	202.07	0.10
(c)	20	20	5	10	0.58	0	10	0.15	0.04	0.36	0.07
(c)	20	20	10	10	0.63	0	10	0.70	0.09	0.59	0.03
(c)	40	40	5	10	24.56	0	10	26.01	0.19	2.38	0.14
(c)	40	40	10	10	15.10	0	10	141.60	0.10	9.29	0.09
(c)	40	40	20	10	10.27	0	8	417.51	0.17	32.37	0.07
(c)	60	60	5	10	784.32	0	10	109.69	0.20	9.30	0.17
(c)	60	60	10	10	218.96	0	5	1178.47	0.19	40.86	0.22
(c)	60	60	20	10	68.13	0	1	1779.41	0.10	173.06	0.07
(c)	80	80	5	0	(*)	0	10	504.18	0.18	30.16	0.21
(c)	80	80	10	4	1641.49	0	0	(*)	0.25	103.15	0.32
(c)	80	80	20	9	537.82	0	0	(*)	0.20	373.26	0.17
(c)	100	100	5	(4) 0	(*)	0	6	800.55	0.17	62.34	0.26
(c)	100	100	10	(0) 0	(*)	n.a.	0	(*)	0	280.35	0.79
(c)	100	100	20	(3) 1	1783.50	0	0	(*)	0.06	1015.64	0.41

Table 5. Random instances - unbalanced case - unit populations

Instance type				B&P			Compact			TS	
version	$\|C\|$	$\|F\|$	H	s	t	g	s	t	g	t	g
(a)	20	10	5	10	0.13	0	10	0.04	0.01	0.05	0
(a)	40	20	5	10	2.11	0.00	10	0.66	0.20	0.47	0.00
(a)	40	20	10	10	1.07	0	10	1.54	0.09	0.60	0.02
(a)	60	30	5	10	14.60	0	10	2.74	0.09	1.24	0.03
(a)	60	30	10	10	9.24	0	10	16.35	0.08	3.77	0.03
(a)	60	30	20	10	4.27	0	10	194.97	0.26	10.03	0.13
(a)	80	40	5	10	84.93	0.00	10	30.79	0.27	4.52	0.02
(a)	80	40	10	10	70.26	0.00	9	184.05	0.15	11.65	0.06
(a)	80	40	20	10	15.62	0	10	126.24	0.03	30.95	0.07
(a)	100	50	5	10	527.81	0.01	10	511.83	0.32	11.99	0.10
(a)	100	50	10	10	282.74	0.01	5	849.75	0.19	38.51	0.05
(a)	100	50	20	10	91.88	0.00	1	1725.29	0.27	110.16	0.19
(b)	20	10	5	10	0.13	0	10	0.04	0.00	0.05	0
(b)	40	20	5	10	2.01	0	10	0.52	0.09	0.42	0.02
(b)	40	20	10	10	1.05	0	10	1.18	0.02	0.63	0.03
(b)	60	30	5	10	14.80	0.00	10	8.85	0.21	1.08	0.07
(b)	60	30	10	10	7.71	0	10	69.04	0.09	3.29	0.02
(b)	60	30	20	10	3.70	0	10	183.35	0.27	9.96	0.19
(b)	80	40	5	10	130.77	0.00	10	88.52	0.31	3.91	0.12
(b)	80	40	10	10	45.80	0.00	9	436.31	0.18	12.56	0.04
(b)	80	40	20	10	12.92	0	10	273.20	0.04	30.45	0.15
(b)	100	50	5	10	399.28	0.01	10	417.80	0.35	9.15	0.14
(b)	100	50	10	10	314.10	0.00	1	122.77	0.21	37.03	0.09
(b)	100	50	20	10	73.02	0.00	1	1733.84	0.30	97.46	0.13
(c)	20	20	5	10	0.53	0	10	0.14	0.08	0.35	0.07
(c)	20	20	10	10	0.48	0	10	0.41	0.03	0.81	0.09
(c)	40	40	5	10	22.03	0	10	27.21	0.19	1.92	0.17
(c)	40	40	10	10	15.34	0	10	87.12	0.10	4.58	0.07
(c)	40	40	20	10	5.40	0.00	10	51.61	0.03	15.41	0.13
(c)	60	60	5	10	747.65	0	10	247.40	0.22	11.40	0.13
(c)	60	60	10	10	208.22	0.00	4	1253.30	0.24	29.78	0.16
(c)	60	60	20	10	35.70	0	1	1545.28	0.10	91.44	0.02
(c)	80	80	5	(9) 0	(*)	0	7	595.64	0.18	28.95	0.14
(c)	80	80	10	6	1301.21	0.00	0	(*)	0.28	106.87	0.28
(c)	80	80	20	10	440.09	0.00	0	(*)	0.17	264.97	0.09
(c)	100	100	5	(4) 0	(*)	0	5	1387.03	0.13	58.64	0.32
(c)	100	100	10	0	(*)	n.a.	0	(*)	0.00	206.11	0.81
(c)	100	100	20	2	1611.00	0.00	0	(*)	0.20	540.32	0.23

Table 6. Random instances - balanced case - unit populations

Instance type				B&P			Compact			TS	
version	$\|C\|$	$\|F\|$	H	s	t	g	s	t	g	t	g
(a)	20	10	5	10	0.19	0	10	0.06	0.02	0.08	0.04
(a)	40	20	5	10	3.51	0	10	0.57	0.05	0.95	0.03
(a)	40	20	10	10	2.10	0	10	2.33	0.17	1.12	0.07
(a)	60	30	5	10	37.25	0.00	10	2.75	0.08	1.23	0.08
(a)	60	30	10	10	19.10	0.00	10	16.06	0.07	6.16	0.04
(a)	60	30	20	10	5.06	0	10	155.33	0.28	21.34	0.21
(a)	80	40	5	10	213.48	0.00	10	22.31	0.22	4.46	0.16
(a)	80	40	10	10	75.58	0.00	10	111.64	0.13	18.99	0.13
(a)	80	40	20	10	29.05	0	8	903.14	0.17	61.57	0.05
(a)	100	50	5	9	1066.26	0.01	10	113.82	0.23	10.88	0.11
(a)	100	50	10	9	454.64	0.00	9	579.17	0.16	57.57	0.13
(a)	100	50	20	10	83.80	0	1	1573.98	0.23	160.24	0.05
(b)	20	10	5	10	0.24	0	10	0.05	0.04	0.08	0.11
(b)	40	20	5	10	3.49	0	10	0.58	0.09	0.88	0.02
(b)	40	20	10	10	2.33	0	10	1.72	0.07	1.12	0.09
(b)	60	30	5	10	29.16	0.00	10	4.56	0.14	1.27	0.10
(b)	60	30	10	10	12.56	0	10	33.92	0.08	6.29	0.06
(b)	60	30	20	10	4.19	0	10	218.89	0.28	21.28	0.10
(b)	80	40	5	10	246.25	0.00	10	24.18	0.24	5.09	0.22
(b)	80	40	10	10	104.62	0.00	9	86.92	0.16	18.01	0.11
(b)	80	40	20	10	26.75	0	6	610.32	0.15	67.67	0.04
(b)	100	50	5	10	603.93	0	10	78.42	0.23	12.02	0.27
(b)	100	50	10	9	307.82	0.00	6	437.44	0.18	51.03	0.16
(b)	100	50	20	10	91.74	0	1	889.35	0.27	209.93	0.10
(c)	20	20	5	10	0.73	0	10	0.15	0.05	0.51	0.14
(c)	20	20	10	10	1.16	0	10	0.73	0.10	0.61	0.04
(c)	40	40	5	10	51.39	0	10	3.57	0.14	2.03	0.14
(c)	40	40	10	10	30.92	0.00	10	82.70	0.09	8.29	0.09
(c)	40	40	20	10	15.45	0.00	9	377.39	0.16	31.01	0.09
(c)	60	60	5	(9) 7	1026.45	0	10	40.69	0.12	10.22	0.16
(c)	60	60	10	10	400.69	0.00	7	365.49	0.17	45.32	0.21
(c)	60	60	20	10	106.07	0.00	4	987.96	0.10	155.59	0.12
(c)	80	80	5	(8) 0	(*)	0.00	8	98.86	0.17	30.29	0.15
(c)	80	80	10	(2) 0	(*)	0	0	(*)	0.06	137.72	0.49
(c)	80	80	20	6	874.73	0.00	0	(*)	0.17	383.49	0.16
(c)	100	100	5	(0) 0	n.a.	n.a.	6	312.02	0.15	68.31	0.28
(c)	100	100	10	(0) 0	n.a.	n.a.	0	(*)	0.01	266.90	0.77
(c)	100	100	20	(0) 0	n.a.	n.a.	(0) 0	(n.a.)	(n.a.)	819.77	1.41

Table 7. Random instances - unbalanced case - different population sizes

Instance type				B&P			Compact			TS	
version	$\|C\|$	$\|F\|$	H	s	t	g	s	t	g	t	g
(a)	20	10	5	10	0.22	0	10	0.06	0.02	0.05	0.04
(a)	40	20	5	10	2.79	0	10	0.91	0.13	0.50	0.02
(a)	40	20	10	10	1.70	0	10	1.77	0.09	0.62	0.11
(a)	60	30	5	10	23.13	0	10	3.38	0.07	1.22	0.02
(a)	60	30	10	10	14.49	0	10	17.78	0.07	3.76	0.14
(a)	60	30	20	10	5.50	0	10	175.15	0.28	9.95	0.05
(a)	80	40	5	10	183.80	0.00	10	25.43	0.25	4.04	0.07
(a)	80	40	10	10	60.21	0	10	131.50	0.16	12.17	0.04
(a)	80	40	20	10	21.63	0	10	250.24	0.06	30.46	0.10
(a)	100	50	5	10	1001.70	0.01	10	200.59	0.28	10.53	0.14
(a)	100	50	10	10	373.65	0.00	8	470.55	0.18	43.22	0.06
(a)	100	50	20	10	104.25	0	1	1449.22	0.24	130.50	0.10
(b)	20	10	5	10	0.20	0	10	0.05	0.00	0.05	0
(b)	40	20	5	10	2.42	0	10	0.62	0.09	0.54	0.02
(b)	40	20	10	10	1.80	0	10	1.29	0.03	0.61	0
(b)	60	30	5	10	40.85	0.01	10	5.64	0.17	1.13	0.10
(b)	60	30	10	10	13.59	0	10	37.63	0.10	3.62	0.02
(b)	60	30	20	10	5.67	0	10	247.71	0.28	9.95	0.13
(b)	80	40	5	10	244.77	0.01	10	28.16	0.28	4.65	0.06
(b)	80	40	10	10	86.47	0.00	9	82.39	0.16	11.41	0.02
(b)	80	40	20	10	19.37	0	9	281.45	0.04	30.21	0.15
(b)	100	50	5	7	940.70	0.01	10	140.40	0.31	10.34	0.09
(b)	100	50	10	9	322.24	0.01	9	700.74	0.20	44.68	0.10
(b)	100	50	20	10	93.94	0	1	1003.83	0.30	108.61	0.14
(c)	20	20	5	10	0.67	0	10	0.16	0.06	0.34	0
(c)	20	20	10	10	0.73	0	10	0.51	0.06	0.62	0.09
(c)	40	40	5	10	42.52	0	10	5.10	0.14	1.86	0.13
(c)	40	40	10	10	52.40	0.00	10	27.54	0.10	5.26	0.07
(c)	40	40	20	10	11.23	0	10	110.94	0.05	16.20	0.15
(c)	60	60	5	9	1188.49	0	10	89.86	0.19	10.72	0.11
(c)	60	60	10	10	350.79	0.00	7	572.96	0.21	37.72	0.21
(c)	60	60	20	10	109.09	0.00	4	1030.58	0.10	98.39	0.03
(c)	80	80	5	(8) 0	(*)	0.00	9	363.20	0.17	34.43	0.17
(c)	80	80	10	(6) 1	1779.12	0	0	(*)	0.13	99.42	0.47
(c)	80	80	20	9	603.46	0.00	0	(*)	0.15	250.05	0.11
(c)	100	100	5	(0) 0	n.a.	n.a.	5	731.22	0.11	78.93	0.35
(c)	100	100	10	(0) 0	n.a.	n.a.	0	(*)	0.01	192.98	0.77
(c)	100	100	20	(5) 1	1351.07	0	(0) 0	n.a.	n.a.	572.78	0.87

Table 8. Random instances - balanced case - different population sizes