A CP/LNS approach for Multi-day Homecare Scheduling Problems

Luca Di Gaspero[1], Tommaso Urli[1]

DIEGM – University of Udine
Via Delle Scienze, 206 - 33100 Udine, Italy
{luca.digaspero|tommaso.urli}@uniud.it

Abstract. Homecare, i.e., supportive care provided at the patients' homes, is established as a prevalent alternative to unnecessary hospitalization or institutional care (e.g., in a rest home or a nursing home). These activities are provided either by healthcare professional or by non-medical caregivers, depending on the patient's needs (e.g., medical care or just instrumental activities of daily living).
In this paper, we consider the problem of scheduling Homecare Activities, that is, determining the caregivers' daily tours and the schedules of the homecare service to patients. We present a Constraint Programming (CP) formulation of the problem and we propose a Large Neighborhood Search method built upon the CP formulation.

1 Introduction

Supportive care at patients' homes is a prevalent alternative to the classical forms of institutional care (e.g., rest homes, nursing homes, hospitals) because it will increase the patient's quality of life while being more cost effective. *Homecare*[1] activities are performed by caregivers who visit the patient's home, carry out their tasks and then travel to the next patient. This specific feature makes the labor organization of homecare activities different from the one arising in institutional care. In particular, the patients' visits could have specific temporal and/or operator requirements which might impose the simultaneous presence of different caregivers at a given place, thus requiring a coordination of the tours.

In this paper we consider the problem of scheduling homecare activities for a time horizon H consisting of h consecutive *days* ($H = \{0, \ldots, h-1\}$). On each day $d \in H$ we have to schedule a set $\mathcal{A}_d = \{0, \ldots, n_d - 1\}$ of *activities* located in a given geographic location (x_a, y_a) (i.e., the patient's home), with a duration d_a and a number m_a of needed caregivers. In addition, specific requirements on the time window $[\sigma_a, \epsilon_a]$ in which the service must be provided can be imposed.

Activities are performed by a set $\mathcal{E} = \{0, \ldots, E-1\}$ of *caregivers*. Each caregiver $e \in \mathcal{E}$ starts and ends his/her tour from a geographic location (x_e, y_e) and can work on a (possibly different) specific time window $[\sigma_{e,d}, \epsilon_{e,d}]$ on each

[1] We do not distinguish here between activity of daily living and medical forms of homecare, however the latter is also known as *Home Healthcare*.

day $d \in H$. Moreover, he/she cannot perform all the possible activities in $a \in \bigcup_{d \in H} \mathcal{A}_d$ but only those which are *compatible* with his/her skills. This is stated by a binary relation $\rho_{e,a}$, whose value is 1 if caregiver e is compatible with activity a. Finally, because of labor regulations, each caregiver e should work at least $\underline{t}_{e,d}$ hours on a given day or he/she should have a day off. Moreover, a caregiver e cannot work for more than k consecutive days and for more than $\overline{t}_{e,d}$ hours on each day. Moderate violations to this latter constraint are usually allowed, but they will count as a overtime work, which has to be penalized.

The problem consists of determining the daily routes of caregivers and the schedules of activities so that the traveling costs and the use of overtime work is minimized. This problem is genuinely hard, being a combination of *employee scheduling* (more precisely *rostering*) and *vehicle routing with time windows* on a multi-period horizon.

It is worth noticing that the basic problem formulation relies on the hypothesis of existence of perfect solutions: i.e., those for which all activities are assigned to a caregiver. Unfortunately, in practice it is not always possible to serve all activities with the available caregivers, therefore in a more realistic formulation we will allow (but highly penalize) solutions in which some activities are left unassigned. The formulation of the problem in this form is the result of the real-world requirements collected and provided to us by EasyStaff, a company specialized in software solutions to timetabling and scheduling problems.

In this paper, we first introduce a Constraint Programming (CP) model of the problem that is based on a vehicle routing formulation. Then we show how we can generally build effective Large Neighborhood Search methods upon this model. Finally, we show with an experimental evaluation on the proposed solution methods on a set of random instances that simulate the structure of real-world Homecare assignment problems.

1.1 Related work

To the best of our knowledge the first approaches to the problem are due to [1] and [2]. In the first paper, a simple scheduling heuristic is employed, whereas in the latter a MIP model has been formulated. Besides these early works, a few other modeling and solution approaches have been proposed for the Homecare problem. An established way, is to look at the problem from a set-partitioning perspective with side constraints [3,4] and solve it through variants of ILP methods (branch-and-price). In particular, Rasmussen et al. [3] focus on different temporal constraints among activities, which generalize the concept of *synchronization* constraints that will be described in Section 2.2.

Other works deal with a variant of the problem that considers multimodal transportation (i.e., car or public transportation), which is solved by hybrid approaches. Bertels and Fahle [5] use a combination of linear programming, constraint programming, and metaheuristics in a flexible tool that will handle multiple variants of constraints. Rendl et al. [6] solve the problem with a hybrid approach that employs CP for generating a valid initial solution to the problem and improves it through a number of different metaheuristics.

Other recent approaches to the problem are due to Trautsamwieser and Hirsch [7], who also employ a VNS-based metaheuristics, and Allaoua et al. [8], who devise a matheuristic approach based on the decomposition of the rostering and the routing parts.

Our work differs from the previous literature mainly because in all the cited paper the authors consider a daily time horizon. To the best of our knowledge [9,10] are the only papers on this topic that considers a periodic problem and they tackle it with an adaptive LNS approach. The main difference w.r.t. these works is that the authors of these two papers focus on stable solutions w.r.t caregivers, i.e., each patient should preferably always be visited by a single caregiver in the week. Moreover the CP model is based on a scheduling perspective and the LNS scheme they use is finer-grained that ours, since they relax a small set of activities either randomly or based on the penalty contribution to the cost function, whereas we consider a coarse-grained neighborhood of one or more days.

Another difference of the present work with previous ones is that in some of the cited papers the authors explicitly model personnel skills, and allow an activity to be performed only by a sufficiently skilled caregiver. In our model, instead, skills are not explicitly considered but they are modeled through incompatibilities between activities and caregivers.

2 A Constraint Model for Homecare Activity Scheduling

The constraint model is based on the classical CP model for the Vehicle Routing Problem (VRP) [11] with some variants needed to capture the special structure of our problem.

We present our model in stages. Let first consider a single day, disregard the temporal components, assume that all homecare activities require exactly one caregiver, and that all activities have to be scheduled. In this setting, we build a special directed graph G, which consists of the following kinds of nodes:

- the departing node for each caregiver,
- the activities that should be executed by the caregivers in the tour,
- the ending nodes for each caregiver.

An edge between two nodes (a_1, a_2) is weighted with the distance of traveling from a_1 to a_2.

The graph G contains $2E + n_d$ nodes, where $E = |\mathcal{E}|$ is the number of caregivers and n_d is the number of activities on day d. This graph structure allows to define *successor* and *predecessor* variables for each node of the graph, which will represent the caregivers routes. By imposing that the successor of the ending node for a given caregiver e is set to be the departing node of the following caregiver $e + 1 \mod E$, the problem consists in finding a Hamiltonian circuit in this graph. In this encoding, the sub-tour for the single caregiver e starts at node $\underline{e} = e$ and ends at node $\overline{e} = E + n_d + e$.

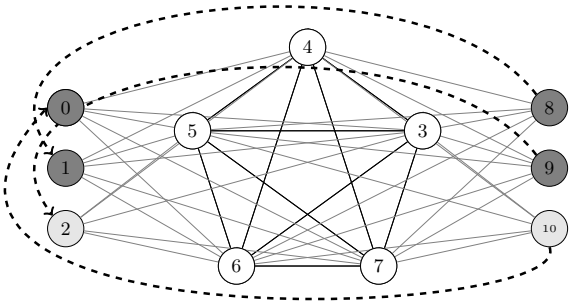


Fig. 1. Graph encoding of the Homecare Activity Scheduling problem employed in this work. The graph considers only one day of the time-horizon, two regular caregivers and five activities.

We first remove the last assumption, i.e., we allow some activities to remain unscheduled. In order to do this, we use an additional "*dummy*" caregiver E, whose sub-path comprises all the activities that are left unserved. Consequently, the set of caregivers becomes $\mathcal{E} = \{0, \ldots, E\}$ and we will denote the set of "*regular*" caregivers by $\mathcal{E}^* = \{0, \ldots, E-1\}$. A pictorial representation of the graph encoding G is illustrated in Figure 1.

Secondly, we can easily remove the restriction on the number of caregivers required by an activity a by adding to the graph one replicate of the activity node for each caregiver required. As a consequence, the graph will have $n_d + r_d$ inner nodes, where r_d is the overall number of activity replicates on day d.

Thirdly, we introduce the time components by extending each node of the graph with a time window. As for the caregiver departing and ending node, the time window will coincide with the allowed worktime of the caregiver, whereas for the activity nodes it will consider the patient's temporal constraints.

In summary, the original daily sub-problem for day d of the time horizon is mapped to the graph G whose vertex set V is partitioned in three sets: $S = \{0, \ldots, E\}$, of departing nodes, $R = \{E{+}1, E{+}2, \ldots, E{+}_{,1}, \ldots, E{+}1{+}(n_d{+}r_d)\}$ of regular activities (with possible replicates), and $T = \{E + 1 + (n_d + r_d) + 1, \ldots, 2(E+1) + (n_d + r_d)\}$ of ending nodes, defined as follows:

The tour of caregiver e starts at node $\underline{e} = e \in S$, proceeds to some "regular" activity nodes belonging to the set R and ends at the ending node $\overline{e} = E + 1 + (n_d + r_d) + e \in T$.

Finally, the extension to multiple days is straightforward, as it will consider a set of distinct graphs, one for each day in the time horizon.

In the following, we give a detailed description of the variables, constraints, and cost function involved in our model. Moreover, we describe a custom branching strategy that we employ to exploit problem-specific knowledge to guide the CP tree-search.

2.1 Variables

Similarly to the previous section, for the purpose of easing notation, the model variables are presented for a single day. The extension to multiple days is straightforward and will be denoted by using an additional superscript index $d \in H$ on all the following variables (which are summarized in Table 1).

As already mentioned, the routes in the graph G are represented by defining the successor of each node in V. Thus, we have $|V|$ successor variables succ that range over V, where succ_i represents the node following node i in the route.

In addition, we define the redundant set of predecessor variables pred where pred_i denotes the node which comes just before node i in the route. Although these variables are redundant, according to [11], when channeled with the previous ones they achieve a more effective propagation.

Second, to each node i we associate the caregiver that will serve that activity[2] through the variable $\mathsf{caregiver}_i$ that ranges over $\mathcal{E}$.

The temporal components are captured by the variables start_i, $\mathsf{duration}_i$, and slack_i, where the first two variables represent the start time and the duration of activity i, whereas the third is an optional waiting time of the caregiver after the activity has been served. In order to deal with manageable domain sizes for these variables, time values have been discretized by expressing them using timeslots of a given time granularity γ. For the purposes of this problem we choose $\gamma =$ 10 minutes, which is adequate to express usual activity duration and practical traveling times, also allowing the possible compensation for small delays. The domain of the time variables for each day d, therefore, are proper subsets of the range $\{0, \ldots, 24\text{h}/\gamma\}$. Concerning the duration variables, in our problem they are statically determined in the problem instance we are considering. However, for modeling purposes we allow also the singular value 0 for duration that, along with the assignment to the dummy caregiver, will represent the duration of unscheduled activities. For this reason, they should be treated as variables. slack variables allow for flexible waiting times in case of spread activity time windows or activity synchronization. The end variables are just convenience variables used for representing the activity end time.

Finally, the variables worktime, overtime, and isWorking are defined on the set of regular caregivers $\mathcal{E}^*$. Variable $\mathsf{worktime}_e$ and $\mathsf{overtime}_e$ accounts for the daily worktime and overtime of caregiver $e \in \mathcal{E}^*$, whereas the boolean variables $\mathsf{isWorking}_e$ state whether the caregiver e is working or has a day off.

The components of the cost function we consider are the number of unscheduled activities, the total overtime (i.e., $\sum_{e \in \mathcal{E}} \mathsf{overtime}_e$) and the total traveled distance. Their costs will be aggregated in a linear combination and expressed in monetary units (namely €).

[2] We consider the starting and the ending nodes as placeholder activities that require exactly that specific caregiver.

Table 1. Variables in the CP Model; all variables are superscripted with the day d of the time horizon which they refer to.

name[dimension]	domain	description
$\mathsf{succ}^d[V]$	V	successor of activity $i \in V$
$\mathsf{pred}^d[V]$	V	predecessor of activity $i \in V$
$\mathsf{caregiver}^d[V]$	$\mathcal{E}\nabla$	caregiver serving activity $i \in V$
$\mathsf{start}^d[V]$	$[0 \dots 24\text{h}/\gamma]$	the starting time of serving activity $i \in V$
$\mathsf{duration}^d[V]$	$\{0, d_i\}$	the duration of activity $i \in V$
$\mathsf{slack}^d[V]$	$[0 \dots 24\text{h}/\gamma]$	possible slack time after serving activity $i \in V$
$\mathsf{end}^d[V]$	$[0 \dots 24\text{h}/\gamma]$	the ending time of serving activity $i \in V$
$\mathsf{worktime}^d[\mathcal{E}]$	$[0 \dots 24\text{h}/\gamma]$	total worktime of caregiver $e \in \mathcal{E}$
$\mathsf{overtime}^d[\mathcal{E}]$	$[0 \dots 24\text{h}/\gamma]$	total overtime of caregiver $e \in \mathcal{E}$
$\mathsf{isWorking}^d[\mathcal{E}]$	$\{0, 1\}$	caregiver is working on this day
$\mathsf{unscheduled}^d$	$[0 \dots n_d]$	number of unscheduled activities
$\mathsf{distance}^d$	$[0, ub]$	total traveled distance
cost^d	$[l, u]$	overall cost

2.2 Constraints

The presentation of the constraints in our model are divided in two parts. First we introduce the essential constraints that are needed to thoroughly model the problem and then we discuss some redundant constraints that will make the tree-search process more efficient by allowing an early detection of infeasible solutions. As in the previous section we temporarily omit the day superscript d on the variables.

Essential constraints. We start our description with the routing part of the model. The first set of constraints will state the relationship between the successor and predecessor variables for all the nodes. This will be achieved by channeling them through the `element` global constraint.

$$\texttt{element}(\mathsf{pred}, \mathsf{succ}_i) = i \quad \forall i \in V \tag{1}$$

$$\texttt{element}(\mathsf{succ}, \mathsf{pred}_i) = i \quad \forall i \in V \tag{2}$$

In order to build a circuit the connections between the ending node of one caregiver and the starting node of the following one are explicitly set.

$$\mathsf{succ}_{E+(n_d+r_d)+e} = (e+1) \mod (E+1) \quad \forall e \in \mathcal{E} \tag{3}$$

$$\mathsf{pred}_e = E + 1 + (n_d + r_d) + (e - 1 \mod (E+1)) \quad \forall e \in \mathcal{E} \tag{4}$$

Finally, the paths described by the succ and pred variables must be Hamiltonian circuits, and this is stated by means of the `circuit` global constraint. The variant of the constraint we use, also binds an array of variables with the distance of the edges selected by the path variables.

$$\texttt{circuit}(\mathsf{succ}, \mathcal{TD}, \mathit{forwardDistance}) \tag{5}$$

$$\texttt{circuit}(\mathsf{pred}, \mathcal{TD}, \mathit{backwardDistance}) \tag{6}$$

In the previous constraints, $\mathcal{TD}$ denotes the matrix with the traveling distances between any pair of nodes in the graph. This is a static data of the problem instance that can be either computed exactly (e.g., through GIS APIs) or just approximated (e.g., considering the haversine distance). Moreover, *forwardDistance* and *backwardDistance* are the arrays with the distances of the selected edges. These arrays cannot be simply summed up in order to compute the total traveled distance because also the path of the dummy caregiver will be included in the circuit. We show in a while a workaround to this situation.

We proceed with constraints on the caregivers' variables. First, we initialize the correct caregiver $e \in \mathcal{E}$ for each of the starting and the ending nodes:

$$\mathsf{caregiver}_e = e \quad \forall e \in \mathcal{E} \tag{7}$$

$$\mathsf{caregiver}_{E+1+(n_d+r_d)+e} = e \quad \forall e \in \mathcal{E} \tag{8}$$

second, we push the caregiver-chain over the path variables on the regular activity nodes so that every node in a path must be served by the same caregiver:

$$\texttt{element}(\mathsf{caregiver}, \mathsf{succ}_i) = \mathsf{caregiver}_i \quad \forall i \in R \tag{9}$$

$$\texttt{element}(\mathsf{caregiver}, \mathsf{pred}_i) = \mathsf{caregiver}_i \quad \forall i \in R \tag{10}$$

third, we remove a given caregiver from the domain of his/her incompatible activities.

$$\rho_{i,e} = 0 \Rightarrow \mathsf{caregiver}_i \neq e \quad \forall i \in R \tag{11}$$

As a result of these constraints, every activity in a path starting from node e and ending on node $E+1+(n_d+r_d)+e$ will be served by the same caregiver e. Consequently, we can filter out the edges traveled by the dummy caregiver by means of a linear combination of the *forwardDistance* and *backwardDistance* variables whose multipliers are the boolean expressions stating whether the outgoing edge from node i is traveled by the dummy caregiver, that is:

$$\mathsf{distance} = \sum_{i \in V} (\mathsf{caregiver}_i = E) \cdot \mathit{forwardDistance}_i \tag{12}$$

Analogously, we can bind the number of activities that are left unscheduled with the following constraint:

$$\texttt{count}(\mathsf{caregiver}, \{E\}, \mathsf{unscheduled}) \tag{13}$$

Next, we present the temporal and scheduling constraints. However, before doing that let us describe some modeling assumption we made in order to have a tractable and uniform expression of these constraints. Since the number of activities that will be assigned to the dummy caregiver E cannot be predicted in advance, we decided to model the time variables of the special path starting from the depot E with "zero" times. That is, we write the constraints that will impose all the start, the duration, and the slack to be 0 on that path. Conversely, on

regular caregivers these variables will be constrained according to the temporal constraints that are statically expressed in the problem instance.

Therefore, the first set of temporal constraints for regular activity nodes will deal with the special case of the dummy caregiver:

$$\mathsf{caregiver}_i = E \Rightarrow \mathsf{start}_i = 0 \wedge \mathsf{slack}_i = 0 \quad \forall i \in R \tag{14}$$

$$\mathsf{caregiver}_i = E \iff \mathsf{duration}_i = 0 \quad \forall i \in R \tag{15}$$

Conversely, for a regular caregiver the following constraints will hold:

$$\mathsf{caregiver}_i \neq E \Rightarrow \mathsf{start}_i \geq \sigma_i \wedge \mathsf{end}_i \leq \epsilon_i \quad \forall i \in V \tag{16}$$

These constraints are also extended on starting and ending nodes of the regular caregivers, where the time window $[\sigma_e, \epsilon_e]$ for those nodes is the working time window for the caregiver $e \neq E$. The variables end are used here to represent the ending time of the activities and they will be constrained as follows:

$$\mathsf{end}_i = \mathsf{start}_i + \mathsf{duration}_i + \mathsf{slack}_i \quad \forall i \in V \tag{17}$$

For starting and ending nodes (S and T), the variables $\mathsf{duration}$ and slack are set to zero:

$$\mathsf{duration}_i = 0 \wedge \mathsf{slack}_i = 0 \quad \forall i \in S \cup T \tag{18}$$

The time-chain on the regular nodes is pushed through the following constraints:

$$\begin{aligned} \mathtt{element}(\mathsf{start}, \mathsf{succ}_i) = \mathsf{start}_i + {} & \\ & (\mathsf{caregiver}_i \neq E) \cdot \big(\mathsf{duration}_i + \mathsf{slack}_i + \\ & \mathtt{element}(\mathcal{TT}, i, \mathsf{succ}_i)\big) \quad \forall i \in S \cup R \end{aligned} \tag{19}$$

where $\mathcal{TT}$ is the matrix of the traveling time between two nodes and the expression $(\mathsf{caregiver}_i \neq E)$ is a boolean multiplier that will propagate the zero start in the case of the dummy caregiver.

Because of the modeling choice on activities requiring multiple caregivers, which are exploded in different replicates of the same activity node, we must ensure that those replicas are synchronized. Moreover, we require that either all the replicates are assigned to a suitable regular caregiver or all replicates remains unscheduled. These conditions are stated by the following constraints:

$$\mathtt{count}(\mathsf{caregiver}_{i,\ldots,i+m_i-1}, \mathcal{E}^*, \mathit{regularCaregivers}) \quad \forall i \in R^* \tag{20}$$

$$\mathtt{nvalues}(\mathsf{start}_{i,\ldots,i+m_i-1}, 1) \quad \forall i \in R^* \tag{21}$$

$$\mathtt{nvalues}(\mathsf{duration}i, \ldots, i + m_i - 1, 1) \quad \forall i \in R^* \tag{22}$$

The set R^* denotes the set of nodes that are the first replicate of an original activity and m_i is the number of caregivers needed. Variable *regularCaregivers* is bound with the number of regular caregivers assigned to the activity replicates and its domain is $\{0, m_i\}$ so that is either zero or the correct number of replicates.

The following two set of constraints impose synchronization of activities start times and duration.

Next, the schedule of the activities assigned to the same regular caregiver must not overlap in time. To this aim we consider each caregiver as a unary resource and the activities as tasks and impose the following scheduling constraints:

$$\texttt{unary}(\mathsf{start}_{i\in R}, \mathsf{end}_{i\in R}, (\mathsf{caregiver}_{i\in R} = e)) \quad \forall e \in \mathcal{E}^* \tag{23}$$

The expression $(\mathsf{caregiver}_{i\in R} = e)$ will be expanded in an array of $|R|$ boolean variables that will activate the no-overlap constraint only on the indexes of the temporal variables for which the expression is true.

Finally, for this family of constraints, the computation of worktime and overtime is given by the following constraints:

$$\mathsf{worktime}_e = \mathsf{start}_{E+(n_d+r_d)+e} - \mathsf{start}_e \quad \forall e \in \mathcal{E}^* \tag{24}$$

$$\mathsf{overtime}_e = \max\{0, \mathsf{worktime}_e - \bar{t}_e\} \quad \forall e \in \mathcal{E}^* \tag{25}$$

Because of the labor regulations concerning the daily worktime, we also impose a minimum amount of worktime $\underline{t}_e$ for caregiver e or we give him/her a day off:

$$\mathsf{worktime}_e \geq \underline{t}_e \vee \mathsf{worktime}_e = 0 \quad \forall e \in \mathcal{E}^* \tag{26}$$

The last family of constraints will deal with the multi-day horizon of our problem. To this aim, we reintroduce from now on the superscript d on variables belonging to different days. The first constraint will bind the $\mathsf{isWorking}$ variables as follows:

$$\mathsf{isWorking}_e^d \iff \mathsf{worktime}_e^d > 0 \quad \forall e \in \mathcal{E}^* \tag{27}$$

According to these variables, the number of consecutive days a caregiver could work can be limited to k by the following sequence constraints:

$$\texttt{sequence}(\mathsf{isWorking}_e^{d\in H}, 1, h, wd) \quad \forall e \in \mathcal{E}^* \tag{28}$$

where wd is a variable whose domain is $\{0, 1, \ldots, k\}$ thus allowing consecutive sequences of at most k ones in the $\mathsf{isWorking}$ variables for caregiver e in the time horizon H.

This concludes the description of the essential of our CP model for the home-care problem. The model can be enhanced by some redundant constraints, that will take care of some particular substructure of the problem and will be presented in the next section together with a custom propagator that will perform a look-ahead so to increase the pruning capabilities for the routing part.

Redundant constraints. The first family of redundant constraints avoid subtours from a starting node to another starting node or from an ending node to another ending node.

$$\mathsf{succ}_i \geq |S| \quad \forall i \in S \cup R \tag{29}$$

$$\mathsf{pred}_i < |S \cup R| \quad \forall i \in S \cup R \tag{30}$$

Moreover, from starting or ending nodes we allow either a path through regular nodes or a short-circuit path from the starting node to the corresponding ending node of a caregiver:

$$\mathsf{succ}_e < |S \cup R| \vee \mathsf{succ}_e = E + 1 + (n_d + r_d) + e \quad \forall e \in \mathcal{E}^* \tag{31}$$

$$\mathsf{pred}_{E+1+(n_d+r_d)+e} < |S \cup R| \vee \mathsf{pred}_{E+1+(n_d+r_d)+e} = e \quad \forall e \in \mathcal{E}^* \tag{32}$$

Similarly to constraint (12), we impose that the computed traveled distance in should be also equal to the backward Hamiltonian circuit (i.e., the path defined through pred variables):

$$\mathsf{distance} = \sum_{i \in V} (\mathsf{caregiver}_i = E) \cdot \mathit{backwardDistance}_i \tag{33}$$

It is possible to filter out edges that cannot belong to any route either by looking at some static problem instance information or at dynamic assignments. As for the static filtering we look at the allowed time windows of pair of activities:

$$\mathsf{caregiver}_i \neq E \wedge [\sigma_i, \epsilon_i] \succ [\sigma_j, \epsilon_j] \Rightarrow \mathsf{succ}_i \neq j \quad \forall i, j \in V \tag{34}$$

The $\succ$ relation between time intervals is true when the left-hand interval do not overlap with the right-hand one and $\sigma_i > \epsilon_j$. The dynamic filtering also looks at pair of activities and their temporal assignments:

$$\mathsf{caregiver}_i \neq E \wedge \mathsf{start}_i > \mathsf{end}_j \Rightarrow \mathsf{succ}_i \neq j \quad \forall i, j \in V \tag{35}$$

For the synchronization among multiple replicates, we also force the following constraint:

$$\begin{aligned} \mathit{regularCaregivers} = 0 \iff \mathsf{caregiver}_i = E \wedge \mathsf{caregiver}_{i+1} = E \wedge \ldots \\ \ldots \wedge \mathsf{caregiver}_{i+m_i} = E \quad \forall i \in R^* \end{aligned} \tag{36}$$

where the *regularCaregivers* is the one that corresponds to constraint (20).

Thanks to the isWorking variables, we can explicit pruning caregiver's values from the domain of regular activities, i.e.:

$$\neg \mathsf{isWorking}_e \Rightarrow \mathsf{caregiver}_i \neq e \quad \forall e \in \mathcal{E}^* \quad \forall i \in R \tag{37}$$

In addition, we can immediately force the short-circuiting of the paths of non working caregivers:

$$\neg \mathsf{isWorking}_e \Rightarrow \mathsf{succ}_e = E + (n_d + r_d) + e \quad \forall e \in \mathcal{E}^* \tag{38}$$

$$\neg \mathsf{isWorking}_e \Rightarrow \mathsf{pred}_{E+(n_d+r_d)+e} = e \quad \forall e \in \mathcal{E}^* \tag{39}$$

and impose a conventional starting time to the departing and ending nodes so to break symmetries:

$$\neg \mathsf{isWorking}_e \iff \mathsf{start}_e = \sigma_e \wedge \mathsf{start}_{E+1+(n_d+r_d)+e} = \sigma_e \quad \forall e \in \mathcal{E}^* \tag{40}$$

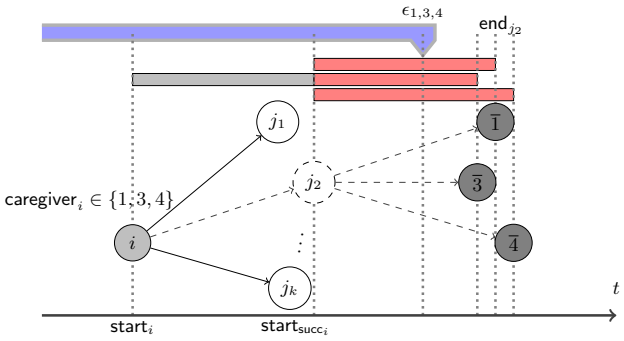


Fig. 2. The look-ahead propagator: value j_2 can be removed from the domain of succ_i if all two steps paths from i through j_2 to a compatible ending node $\overline{e}$ exceed the corresponding time window.

Look-ahead propagator. The final redundant constraint we impose is a look-ahead constraint. This constraint has been implemented by means of a custom propagator that will perform a one-step look-ahead of the temporal variables of the successors of a given node. The idea of this propagator (see Figure 2) is to prune a value j from the succ_i variable if all two-step-paths from i to the ending depot d_e passing through j will violate the temporal constraints (16).

Cost function. The cost function of the problem is a hierarchical one, and comprises the three components measured by the variables unscheduled, distance, and overtime. Moreover, since we would like to maximize the efficient use of caregivers' worktime we also penalize the use slack in solutions.

We consider a weighted aggregation of these components to obtain a measure in a common unit of measurement, i.e.:

$$\begin{aligned}\mathsf{cost} = w_1 \sum_{d \in H} \mathsf{unscheduled}^d + w_2 \sum_{d \in H} \sum_{e \in \in \mathcal{E}^*} \mathsf{overtime}_e^d + \\ w_3 \sum_{d \in H} \sum_{i \in V^d} \mathsf{slack}_i^d + w_4 \sum_{d \in H} \mathsf{distance}^d\end{aligned} \tag{41}$$

where $w_1 = 100€$, $w_2 = w_3 = 25€/\text{h}$, and $w_4 = 0.30€/\text{Km}$. These weight values have been set up according to the current work and traveling costs.

2.3 Branching strategy.

The branching strategy we employ is a bi-level one. First we determine the number of caregivers that will work on a given day of the timeslot. This is done through a surrogate counting variable that sums up the values of the isWorking boolean variables. The heuristic for value selection for this brancher is to take

the median value. The motivation is that we have also to assign days off to caregivers, therefore it would be effective to spread these days off on the whole time horizon. Once the number of available caregivers have been set, a second brancher will randomly select the value of the isWorking variables accordingly.

In the second level of the branching strategy we aim at constructing the caregivers' routes and determine the activity schedule. To do so, in order to exploit the problem structure, we implemented a custom branching strategy, whose behavior is illustrated in Figure 3. The procedure tries to extend the shortest caregivers' route (Fig. 3(a)) by setting the succ variable of its current last step. The selection of the next activity is driven by the following heuristics (in order of importance):

1. prefer a regular node over an ending node;
2. if replicates of multiple activities have already been scheduled, prefer an unscheduled replicate of those activities;
3. prefer an activity that has to end earlier;
4. prefer shorter activities;
5. prefer an activity that has to start earlier;
6. prefer the activity with most replicates.

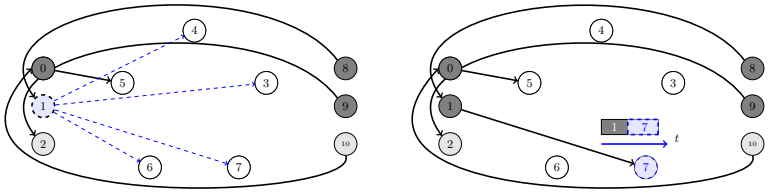
Once the succ variable has been selected, the brancher will set the start variable of the just added node to its earliest value (Fig. 3(b)). All the other temporal variables will be fixed thanks to constraint propagation. Once the routes of regular caregivers are closed (i.e., they reach the ending node), all the remaining activities are assigned to the dummy caregiver (Fig. 3(c)) and their temporal variables are assigned to the conventional value 0 (Fig. 3(d)).

3 Hybrid approach

Large Neighborhood Search (LNS) [12,13] is a neighborhood search meta-heuristic based on the observation that exploring a *large neighborhood*, i.e., perturbing a significant portion of a solution, generally leads to optima of much *higher quality*. While this is an undoubted advantage in terms of search, exploring a large neighborhood structure can be *computationally impractical*, and requires a higher effort than exploring of a regular neighborhood. For this reason, LNS has been often coupled with filtering techniques, with the aim of reducing the size of the neighborhood by neglecting those choices that would lead to unfeasible solutions. In particular, LNS has been often associated with constraint models in order to tackle complex routing problems [14,15]. In the following we describe our LNS approach.

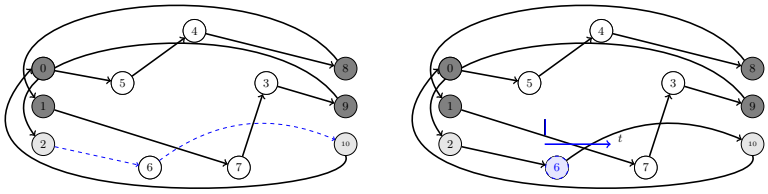
The LNS procedure is initialized by generating the first feasible solution using the branching strategy described above. This step is essentially equivalent to finding the first feasible solution with our CP model.

Once the initial solution has been generated, the algorithm enters a refinement loop, which consists of two alternate steps. First, the *destroy* step, relaxes



(a) Extend the shortest route by setting the succ variable of its last node.

(b) Set the start variable of the activity just added to its earliest value.



(c) Assign remaining activities to the dummy caregiver.

(d) Set their temporal variables to the conventional value 0.

Fig. 3. Custom tree-search branching strategy for the Homecare problem.

(unassigns) a subset of the decision variables, yielding a smaller constrained optimization problem (less variables, filtered domains). Second, the *repair* step, re-optimizes the relaxed variables by means of a tree-search.

Variable relaxation. In our destroy step, a fraction of the decision variables is relaxed uniformly at random based on a parameter $\delta \in \{1, \ldots, h\}$, which represents the number of full days that are relaxed by the destroy step, i.e., if $\delta = 1$ then all the variables related to a single day are relaxed, if $\delta = h$ the solution is completely relaxed. Since the pure CP tree-search assigns the variables day after day, in this understanding this relaxation scheme allows to undo possibly bad decision made by CP towards the root of the search tree.

Re-optimization. Once a subset of the decision variables have been relaxed, a new solution is produced through tree-search. Ideally, the repair step should return the best solution in the neighborhood. However, depending on the number of relaxed variables, finding the optimum can be non-viable. Our re-optimization strategy consists in giving the repair step a time budget, which is proportional to the number of relaxed variables, i.e., a fixed number of milliseconds t_{var}. Moreover, this tree-search chooses the number and identity of the working caregivers uniformly at random.

Destruction rate adaptation Our approach involves a strategy to adapt δ dynamically during the search, starting from $\delta = 1$. Once a maximum number of non-improving iterations ii_{max} has been spent on a certain value of δ, this value

is increased by one. The rationale behind this is that, if the search cannot improve anymore by relaxing one day, then we try relaxing to days at once, and so on. When LNS has performed ii_{max} iterations on $\delta = h$ (the maximum number of relaxable days) the search is restarted.

Solution acceptance A neighboring solution is accepted if its cost is lower or equal to the one of the current best solution. This allows to escape possible plateaux in the fitness landscape.

4 Experimental Analysis

To assess the performance of our hybrid approaches, we have generated a number of random instances based on the presented problem formulation. The behavior of our instance generator, which is publicly available at `https://bitbucket.org/tunnuz/homecare-instance-generator`, is influenced by a number of parameters. Among these are the geographical area, the planning horizon in days, the number of daily activities, a caregiver correction rate (a multiplier that can be used to increase or reduce the number of needed caregivers, which is computed heuristically), the types and probabilities of shifts for caregivers, the probability of a caregiver/activity incompatibility, the probability of an activity to need multiple caregivers, and the parameter k for the Poisson distribution from which the number of caregivers for an activity is drawn.

Overall, our benchmark set is composed of 18 families of 30 instances each (totaling 540 instances), differentiated by planning horizon, number of activities, and caregiver correction rate (c). In these instances, whose activities are located in a 40Km $\times$ 40Km rectangular area centered on Udine (Italy), the number of maximum consecutive days is always $h - 1$ (thus prescribing at least a free day for each caregiver), while the other parameters have their default values. According to the best practices, we have tuned the parameters of our LNS approach ($ii_{max} = 60$ and $t_{var} = 10$ms) by running, through JSON2RUN [16], an *F-Race(RSD)* [17] over a subset of 360 instances, and used the remaining 180 for validation against the pure CP approach. Both the training and validation instances are distributed together with the instance generator.

Table 2 summarizes, for each family of the validation instances, the aggregated mean cost (f) in €, number of unassigned activities (f_u), distance in Kilometers (f_d), hours of overtime (f_o), and hours of slack time (f_s) attained, respectively, by pure CP and our hybrid approach in 10 minutes. The last column shows the cost improvement attained when using LNS instead of pure CP.

From the results, it is easy to see that the LNS approach outperforms pure CP on all except one family of instances, mostly because of its superiority in dealing with unassigned activities (the most weighted cost component) and overtime work. On the other hand, CP seems to be better at reducing the total traveling distance. This is likely due to our branching strategy, which is able to reconsider the caregivers' routes, exploiting the fact that it does not have a bound on the time, unlike LNS repair step.

Table 2. Comparison between performance of CP (branch & bound) and LNS.

Group			CP					LNS					Imp.
h	n_d	**c**	**f**	f_u	f_d	f_o	f_s	**f**	f_u	f_d	f_o	f_s	
3	20	0.6	2247.46	11	399.40	0.32	0.50	**961.37**	2	419.30	0.10	0.30	57.2%
3	20	0.7	1210.38	0	478.20	0.15	0.67	**1059.02**	0	367.80	0.04	0.72	12.5%
3	20	0.8	1358.65	4	345.17	0.14	0.53	**975.08**	0	341.20	0.06	0.64	28.2%
3	30	0.6	2621.19	9	512.43	0.41	0.90	**1220.18**	0	773.53	0.13	0.42	53.5%
3	30	0.7	2908.59	12	538.43	0.28	1.00	**1480.50**	0	591.67	0.27	0.72	49.1%
3	30	0.8	2255.51	9	537.23	0.13	0.78	**1265.01**	0	538.90	0.08	0.73	43.9%
3	40	0.6	4502.60	26	780.67	0.53	0.73	**1851.14**	0	807.93	0.18	0.99	58.9%
3	40	0.7	**2710.03**	4	833.37	0.58	1.05	2968.33	7	867.03	0.45	1.10	-9.5%
3	40	0.8	3330.83	7	785.37	0.52	1.48	**2405.65**	3	948.50	0.33	0.97	27.8%
6	20	0.6	3846.61	19	696.23	0.35	1.03	**2017.17**	0	787.97	0.10	1.27	47.6%
6	20	0.7	2713.60	5	824.00	0.32	1.21	**1605.11**	0	734.57	0.11	0.87	40.8%
6	20	0.8	3088.61	6	836.23	0.42	1.39	**1589.25**	0	725.83	0.08	0.90	48.5%
6	30	0.6	5862.63	27	1114.03	0.76	1.49	**2188.90**	0	1147.67	0.31	0.90	62.7%
6	30	0.7	4371.72	5	1137.47	0.85	2.12	**2922.87**	0	1060.97	0.40	1.65	33.1%
6	30	0.8	4000.81	5	1160.90	0.38	2.18	**2595.11**	1	1154.57	0.33	1.18	35.1%
6	40	0.6	7646.02	36	1557.80	1.02	1.73	**4096.22**	0	1542.47	0.57	2.25	46.4%
6	40	0.7	6721.86	17	1562.07	0.90	2.86	**4827.88**	0	1693.20	0.79	2.65	28.2%
6	40	0.8	6623.41	11	1634.90	0.93	3.30	**5486.30**	7	1540.33	0.73	2.81	17.2%

5 Conclusions and Future Work

In this paper we tackle the problem of multi-day scheduling of homecare activities by means of two CP-based solution methods. The contributions of the paper are the following ones. First, we propose an effective routing-based CP-model for the problem, including custom look-ahead constraints, and a dedicated branching strategy. Second, we devise an adaptive LNS method, which considerably improves the results found by pure CP tree-search.

Among the alternatives we want to explore, are different relaxation methods for the LNS destroy step, and more refined branching strategies either for the determination of caregivers' days off or for a finer-grained construction of the routes. Moreover, we plan to test future refinements of this approach over real-world instances as soon as they will be provided by our industrial partner.

Acknowledgments. This work has been partially funded by Google Inc. under the Google Focused Grant Program on "Mathematical Optimization and Combinatorial Optimization in Europe". We thank EasyStaff, for providing us with the real-world problem specification.

References

1. Begur, S.V., Miller, D.M., Weaver, J.R.: An integrated spatial DSS for scheduling and routing home-health-care nurses. Interfaces **27**(4) (1997) 35–48
2. Cheng, E., Rich, J.L.: A home health care routing and scheduling problem. Technical Report CAAM TR98–04, Rice University (1998)
3. Rasmussen, M.S., Justesen, T., Dohn, A., Larsen, J.: The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies. European Journal of Operational Research **219**(3) (2012) 598 – 610
4. Eveborn, P., Flisberg, P., Rönnqvist, M.: Laps care—an operational system for staff planning of home care. European Journal of Operational Research **171**(3) (2006) 962 – 976
5. Bertels, S., Fahle, T.: A hybrid setup for a hybrid scenario: Combining heuristics for the home health care problem. Computers and Operations Research **33**(10) (2006) 2866–2890
6. Rendl, A., Prandtstetter, M., Hiermann, G., Puchinger, J., Raidl, G.: Hybrid heuristics for multimodal homecare scheduling. In Beldiceanu, N., Jussien, N., Pinson, E., eds.: Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems. Volume 7298 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 339–355
7. Trautsamwieser, A., Hirsch, P.: Optimization of daily scheduling for home health care services. Journal of Applied Operational Research **3** (2011) 124–136
8. Allaoua, H., Borne, S.: A matheuristic approach for solving a home health care problem. Electronic Notes in Discrete Mathematics **41** (2013) 471–478
9. Steeg, J., Schröder, M.: A hybrid approach to solve the periodic home health care problem. In Kalcsics, J., Nickel, S., eds.: Operations Research Proceedings 2007. Volume 2007 of Operations Research Proceedings. Springer Berlin Heidelberg (2008) 297–302
10. Nickel, S., Schröder, M., Steeg, J.: Mid-term and short-term planning support for home health care services. European Journal of Operational Research **219**(3) (2012) 574 – 587
11. Kilby, P., Shaw, P.: Handbook of Constraint Programming: Vehicle Routing. Elsevier, New York, NY, USA (2006)
12. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In J. Maher, M., Puget, J.F., eds.: CP'98: the 4th International Conference on Principles and Practice of Constraint Programming, 1998, Proceedings of. Volume 1520 of Lecture Notes in Computer Science., Springer (1998) 417–431
13. Pisinger, D., Ropke, S.: Large neighborhood search. In: Handbook of Metaheuristics. Springer (2010) 399–419
14. Bent, R., Van Hentenryck, P.: A two-stage hybrid local search for the vehicle routing problem with time windows. Transportation Science **38**(4) (2004) 515–530
15. Rousseau, L.M., Gendreau, M., Pesant, G.: Using constraint-based operators to solve the vehicle routing problem with time windows. Journal of Heuristics **8**(1) (2002) 43–58
16. Urli, T.: json2run: a tool for experiment design & analysis. CoRR **abs/1305.1112** (2013)
17. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-Race and iterated F-race: An overview. Springer, Berlin (2010)