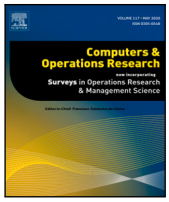
Computers & Operations Research 159 (2023) 106351

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor



An Adaptive Large Neighbourhood Search algorithm for a real-world Home Care Scheduling Problem with time windows and dynamic breaks

Isabel Méndez-Fernández [a,*], Silvia Lorenzo-Freire [a,b], Ángel Manuel González-Rueda [a]

[a] MODES Research Group, Department of Mathematics and Centre for Information and Communications Technology Research (CITIC), Faculty of Computer Science, University of A Coruña, Campus de Elviña, A Coruña, Spain
[b] CITMAga, 15782 Santiago de Compostela, Spain

ARTICLE INFO

Keywords:
Home care
Scheduling
Adaptive large neighbourhood search
Mixed integer linear programming

ABSTRACT

This paper presents a Home Care Scheduling Problem (from now on HCSP) based on a real case of a care company for elderly and dependent people located in the North of Spain. The problem incorporates many of the common features addressed in the HCSP literature, such as soft and hard time windows for the services, available working time of the caregivers or affinity levels between users and caregivers. However, it also includes other novel characteristics that increase the difficulty of the problem significantly, since the breaks between services will play a key role in the quality of the solutions. To evaluate the solutions, the users welfare will be prioritized over the cost associated with the schedule.

The problem has been formulated as a Mixed Integer Linear Programming (MILP) one but, due to the complexity of the model, it is not possible to solve it for real size instances. Therefore, we have designed a method that combines the Adaptive Large Neighbourhood Search (ALNS) methodology with a heuristic approach necessary to evaluate the objective functions. To analyse the behaviour of the algorithm, a set of computational experiments are carried out under different configurations. First, the MILP formulation and the algorithm have been compared over some standard instances from the literature. Finally, the performance of the algorithm is evaluated over a real case study based on the timetables of the company during some weeks from 2016 to 2017.

1. Introduction

A Home Care Scheduling Problem (HCSP) deals with real life situations where some users require home assistance services and they should be attended by a set of caregivers throughout the week. Thus, the goal of an HCSP is to design the routes and schedules for caregivers, indicating the services to carry out, in which order and at what time. Schedules should also satisfy all the requirements imposed by the users and the caregivers' collective labour agreement, while pursuing a set of objectives.

Due to the increase in demand for home care services in recent years, the literature on this type of problems has grown significantly. Cissé et al. (2017) and Fikar and Hirsch (2017) are two interesting surveys that analyse the problem in depth, as well as some of its variants. Table 1 shows an overview of the main characteristics of some HCSP studied in the literature: time windows, the location at which the route starts, fixed or not fixed duration of the services, the compatibilities caregiver–user and the continuity of care. The existence of time windows, the starting point of the routes and the duration of the services are quite common concepts in the context of routing problems. The continuity of care is a concept that appears in some contributions related to the HCSP, although the interpretation of the term depends on the context of the problem to be addressed. So, for instance, Bachouch et al. (2011) consider that all services demanded by a user during the week must be carried out by the same employee. In turn, Cappanera and Scutellà (2015) argue that a certain number of caregivers should not be exceeded while attending a given user during each time period. A similar procedure is followed in Nickel et al. (2012), where the continuity of care is now taken into account in the objective function by means of the nurse–patient loyalty, which is just the number of nurses who have attended the patient over a time horizon minus one. Carello and Lanzarone (2014) introduce different continuity of care requirements, depending on the user and on his/her requests, partitioning the set of users into five subsets. In other HCSP, the compatibility caregiver–user is a crucial aspect in determining the

* Correspondence to: Postal code: 15071, A Coruña, Spain.
E-mail addresses: isabel.mendez.fernandez@udc.es (I. Méndez-Fernández), silvia.lorenzo@udc.es (S. Lorenzo-Freire), angel.manuel.rueda@udc.es (Á.M. González-Rueda).

https://doi.org/10.1016/j.cor.2023.106351
Received 12 April 2022; Received in revised form 7 July 2023; Accepted 7 July 2023
Available online 19 July 2023

Table 1
Brief summary of the HCSP characteristics in the literature.

Reference	Services hard time windows	Services soft time windows	Caregivers time windows	Routes start location	Fixed services durations	Compatibility	Continuity of care	Solution method
Akjiratikarl et al. (2007)	X	–	–	Home	X	–	–	PSO
Bachouch et al. (2011)	X	–	X	Office	X	X	X	LP
Bard et al. (2014)	X	–	X	Various	X	–	–	GRASP
Bertels and Fahle (2006)	X	X	X	First visit	X	X	X	LP+CP+SA/TS
Braekers et al. (2016)	X	X	X	Various	X	X	–	MDLS+LNS
Cappanera and Scutellà (2015)	–	–	X	Various	X	X	X	Heuristic, LP
Carello and Lanzarone (2014)	–	–	–	First visit	–	X	X	LP
Chaieb et al. (2020)	X	–	X	Various	X	X	–	k-means+Hungarian+TS
Decerle et al. (2019)	–	X	X	Office	X	X	–	MA, ACO, MA+ACO
Erdem and Bulkan (2017)	X	X	X	Home	X	X	–	VND
Garaix et al. (2018)	X	–	X	Various	X	X	–	LP
Grenouilleau et al. (2019)	X	–	X	Home	X	X	X	LNS+SPP
Kergosien et al. (2009)	X	–	X	Various	X	X	–	LP
Liu et al. (2017)	X	–	X	Various	X	X	X	B&P+TS
Mankowska et al. (2014)	X (Start)	X (End)	X	Office	X	–	–	LS, AVNS
Maya Duque et al. (2015)	–	X	X	First visit	X	X	–	Heuristic
Méndez-Fernández et al. (2020)	X	–	–	First visit	X	X	X	SA
Mosquera et al. (2019)	X	–	X	Various	–	X	–	LS
Nickel et al. (2012)	X	–	X	Office	X	X	X	ALNS, TS, CP
Rest and Hirsch (2016)	X	–	X	First visit	X	X	–	TS
Riazi et al. (2019)	X	–	X	Various	X	X	–	CG, Gossip+CG
Trautsamwieser and Hirsch (2011)	X	X	X	Various	X	X	–	VNS
Our approach	X	X	X	First visit	X	X	X	ALNS

PSO = Particle Swarm Optimization, GRASP = Greedy Randomized Adaptive Search Procedure, MA = Memetic Algorithm, ACO = Ant Colony Optimization, MDLS = Multi-Directional Local Search, LNS = Large Neighbourhood Search, SPP = Set Partitioning Problem, CG = Column Generation, LS = Local Search, VND = Variable Neighbourhood Descent, VNS = Variable Neighbourhood Search, TS = Tabu Search, B&P = Branch and Price, LP = Linear Programming, CP = Constraint Programming, AVNS = Adaptive Variable Neighbourhood Search, SA = Simulated Annealing, ALNS = Adaptive Large Neighbourhood Search.

schedules. This is the case of Maya Duque et al. (2015), where the compatibility is established by means of several factors, such as the skills of the caregiver, the user's medical status and disabilities, the proficiency of the caregiver in speaking different languages, etc.

Much of the work on HCSP is inspired by real applications. Thus, in Akjiratikarl et al. (2007) a real situation arising in the UK is studied, in collaboration with The Welsh Systems Consortium, a partnership between seven local government authorities in Wales. The work of Grenouilleau et al. (2019) is the result of a joint project with Alayacare, a Canadian start-up developing an operations management platform for home health care agencies. Maya Duque et al. (2015) explores the home care services in Belgium, taking into consideration the case of Landelijke Thuiszorg, an organization that provides home care services in four Belgian regions. The research in Rest and Hirsch (2016) is based on a collaborative project with the Austrian Red Cross, one of the leading home care service providers in Austria.

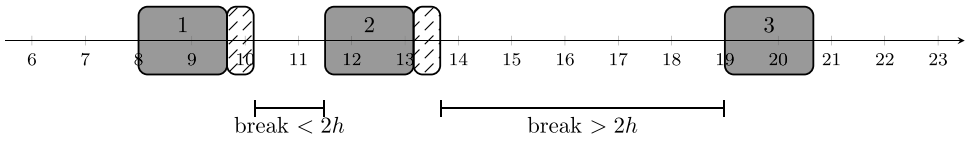
Since the HCSP are a kind of routing and scheduling problems, one of the aspects that is usually assessed in the objective functions of the HCSP is the travelling cost. However, although the minimization of the travelling cost is the only objective in Akjiratikarl et al. (2007) , Bachouch et al. (2011), Kergosien et al. (2009) and Riazi et al. (2019), in many other frameworks this objective is usually accompanied by other targets specific to this type of problems: the minimization of the penalization of services soft time windows (as in Bertels and Fahle, 2006; Braekers et al., 2016; Decerle et al., 2019; Maya Duque et al., 2015), the minimization of overtime costs (considered in Braekers et al., 2016; Cappanera and Scutellà, 2015; Carello and Lanzarone, 2014; Erdem and Bulkan, 2017; Grenouilleau et al., 2019; Nickel et al., 2012; Rest and Hirsch, 2016), the maximization of the continuity of care (studied in Carello and Lanzarone, 2014; Grenouilleau et al., 2019; Nickel et al., 2012), or the maximization of the compatibilities caregiver–user (as in Bertels and Fahle, 2006; Chaieb et al., 2020; Garaix et al., 2018; Maya Duque et al., 2015), among others.

This framework is based on a real case study related to a company located in the Northwest of Spain, which provides home care services for elderly and people with disabilities. The problem under study includes all the characteristics mentioned in Table 1. According to the indications of the staff of the company, based on their previous experiences and requirements, the compatibility caregiver–user and the continuity of care are combined, measuring their importance by means of several levels, which indicate the grade of affinity between caregivers and users. Moreover, a special feature is added to the model: the largest break of a caregiver in a workday, if it reaches a certain duration, will be discounted from the worked time.

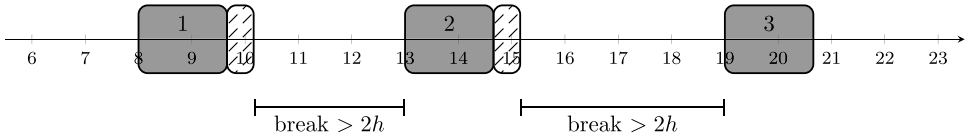
In some HCSP, there is a fixed break which is just a specific amount of time set aside for the caregiver's meal. This is the case of Bard et al. (2014), that considers an uncompensated lunch break of half an hour between 11:00 a.m. and 1:00 p.m., in case the shift is 6 or more hours. In Bachouch et al. (2011), the break is modelled as a preassigned care of one hour within a time window. A different approach is followed in Trautsamwieser and Hirsch (2011), where the duration of the break and its time window are specified by the caregiver. In all these works, the break can be interpreted as a service with a known duration and a time window. The meaning of break is drastically different in this framework, since the duration of the break is not previously fixed and depends on the final schedule of the caregiver. Note that, as mentioned in the example in the Appendix, there may be caregivers with morning shifts, afternoon shifts, or split shifts, which greatly affects the break that may occur. This *dynamic break* may not even exist if there is no rest between consecutive services for more than a specified duration. Moreover, the dynamic break will have a significant impact on the cost of caregivers, since it will be deducted from their daily working times. This particular feature complicates the study of the model considerably.

With all the characteristics described above, the problem under study has been modelled as a bi-objective HCSP, where the objectives are the schedules cost and the users welfare. The schedules cost is related to the overtime and total working time of caregivers. On the other hand, the welfare function collects the affinity between users and caregivers and the soft time windows penalization. According to the specifications of the company, the welfare of the users is prioritized over the cost. In fact, two possible situations can arise when the company starts the planning:

- Let us suppose that there are no historical data about the services schedule. In this case, the welfare of the users will be determined by the soft time windows, that will indicate the periods in which the users would like to be attended.



(a) The only break larger than 2 hours is not considered as working time.



(b) The largest break is not considered as working time.

Fig. 1. Types of breaks.

- In most of cases, the welfare will be obtained from the recent information about the services of the users. Now, the affinity will depend on the users–caregivers assignments, whereas the soft time windows will coincide with the previous schedules as far as possible.

Notice that, although Méndez-Fernández et al. (2020) also deal with the problem of obtaining schedules for the same company, this new framework differs substantially from that one. The only goal of the problem studied in Méndez-Fernández et al. (2020) consisted in intelligently adapting the current schedules to small changes that arose during the week, regardless of the optimality of the solutions. To carry out this process, a simulated annealing algorithm was developed, which led to small delays or advances in the prearranged services.

The present work is the result of a much more ambitious plan for the company. After discussions with the company's managers, it was necessary to consider time windows indicating the caregivers availability and soft time windows for the services. Now, the soft time windows could coincide with the previous schedule of the service or not and their duration could be bigger than the service duration, allowing more flexibility. In this way, schedules are optimized from scratch, considering the hard time windows of the services but also a soft time window that indicates the interval where the users would like to be attended (this information was also provided by the company and, in most of the cases, the soft time windows were different from the previous schedule obtained by the company). Since the preferred time windows for the users could be very different from the previous schedules of the company, there is more room to look for schedules in which overall satisfaction will increase, while improving the global cost of the company.

To solve the problem, an MILP, that adequately describes all the particular features of the problem, is described. However, due to its complexity, a state of the art MILP solver can only be employed to solve small-scale instances.

Thus, to solve real data files, it was necessary to develop a new algorithm, which involves two procedures. The first one is related to generate routes and involves an ALNS methodology based on four insertion and five removal operators to improve the routes iteratively. The second part is the more innovative and is the key element of the algorithm, since it is necessary to compare the routes in terms of the objective functions. This part is devoted to obtain the schedules for the routes. The difficulty of this task lies on the fact it is necessary to combine the hard time windows with the soft time windows to establish the dynamic breaks. Notice that, in the literature related to classical HCSP, the beginning time of the services only depends on the time windows and the starting time of the route. However, this is not the case in this framework, since the break between services causes that there may be multiple different starting times of each service to be combined in order to find the best schedule.

The paper is structured in three sections. Section 2 presents a detailed description of the HSCP, as well as its MILP formulation. Section 3 presents the tailored heuristic algorithm based on the ALNS approach developed to solve the problem. Finally, Section 4 covers the computational experiments used to study the algorithm behaviour.

2. Problem description

Although this problem shares the common features of an HCSP, there are some specific ones that must be taken into account.

The company allows users to indicate two types of time windows: a hard and a soft one. On one hand, the hard time window of a service is the period in which is mandatory to perform the service. On the other hand, the soft time window specifies the time interval where it would be preferable for the service to be performed. Notice that it is not necessary to fulfil the soft time windows to obtain a feasible plan.

According to the contract of each caregiver, a limited number of working hours per day and week are provided. Even so, the total number of hours per week could be increased, as long as they were economically compensated and the limit of daily hours was respected.

In order to take care of users satisfaction, six affinity levels are considered in terms of two features, the skills of the caregiver and the degree of compatibility between users and caregivers. These levels are:

Level 0 The user should not be attended by this caregiver under any circumstances.

Level 1 The caregiver could attend the user only if there is no other option.

Level 2 The caregiver could attend the user although the degree of compatibility between user and caregiver has not been established yet.

Level 3 The caregiver has not attended the user yet but, due to the characteristics of the caregiver, the supervisor thinks that she could be a very good candidate to perform these services.

Level 4 The user was successfully attended by the caregiver in the past.

Level 5 The caregiver is already attending the user adequately.

Each caregiver working day starts at the beginning of the first service and ends when the last service is finished. All the breaks between services will be considered as paid working time with just one exception: in case the largest break of the working day has a duration greater than or equal to π_{min} it will not be paid.

The company currently considers that the minimum duration for the break to not be considered as worked time is 2 h. Thus, in the schedule of Fig. 1(a), where the services duration is 100 min and the travel time between each pair of services is 30 min (represented with hatched blocks), only the largest break (with a duration of 5 h and

Table 2
Sets, parameters and variables involved in the problem.

Sets	
$D = \{1, \dots, 7\}$	Set of days.
$N = \{1, \dots, n\}$	Set of caregivers.
$S = \{1, \dots, s-1\}$	Set of services.
$S^0 = S \cup \{0\}$	Set of services and the initial dummy.
$S^1 = S \cup \{s\}$	Set of services and the ending dummy.
$S^{01} = S \cup \{0, s\}$	Set of services and the initial and ending dummies.
$S_{-k} = S \setminus \{k\}$	Set of services except $k \in S$. Analogously, $S^0_{-k} = S^0 \setminus \{k\}$ and $S^1_{-k} = S^1 \setminus \{k\}$.
Data	
ρ^i_j	It indicates if caregiver $i \in N$ can perform service $j \in S$.
λ^i_j	Affinity level between caregiver $i \in N$ and service $j \in S$.
η_j	Duration of service $j \in S$.
$[\underline{\alpha}^d_j, \bar{\alpha}^d_j]$	Hard time window of service $j \in S$ in day $d \in D$. Note that, if service $j \in S$ does not belong to day $d \in D$, we set $\underline{\alpha}^d_j = \bar{\alpha}^d_j$.
$[\underline{\beta}^d_j, \bar{\beta}^d_j]$	Soft time window of service $j \in S$ in day $d \in D$.
$[\underline{\gamma}^{id}, \bar{\gamma}^{id}]$	Availability time period of caregiver $i \in N$ at day $d \in D$.
θ_{jk}	Travel time between services $j \in S$ and $k \in S^1$. Note that, if $k = s$, then $\theta_{js} = 0$.
ν^i	Agreed weekly working time of caregiver $i \in N$.
ν^{id}	Maximum time caregiver $i \in N$ is allowed to work at day $d \in D$.
π_{min}	Minimum length of time required for the largest break to be unpaid.
Variables	
$x^{id}_{jk} \in \{0, 1\}$	It indicates if caregiver $i \in N$ goes from service $j \in S^0$ to service $k \in S^1$ at day $d \in D$.
$t^{id}_j \in \mathbb{R}^+_0$	For caregiver $i \in N$, it represents the starting time of service $j \in S^{01}$ at day $d \in D$.
$y^{id}_{jk} \in \{0, 1\}$	For caregiver $i \in N$, it indicates if the break between services $j \in S$ and $k \in S$ has been selected to be discounted from the working day $d \in D$.
$\bar{y}^{id} \in \{0, 1\}$	It states if there is no break for caregiver $i \in N$ at day $d \in D$.
$r^{id} \in \mathbb{R}^+_0$	Greatest break of caregiver $i \in N$ at day $d \in D$.
$u^{id} \in \{0, 1\}$	It indicates if the largest break of caregiver $i \in N$ at day $d \in D$ is greater than or equal to π_{min}.
$\hat{r}^{id} \in \mathbb{R}^+_0$	Greatest break of caregiver $i \in N$ at day $d \in D$ if it is greater than or equal to π_{min}. Otherwise, it will be 0.
$z^i \in \mathbb{R}^+_0$	Amount of overtime of caregiver $i \in N$.
$v^{start}_j \in \mathbb{R}^+_0$	Penalization for carrying out service $j \in S$ before its soft time window.
$v^{end}_j \in \mathbb{R}^+_0$	Penalization for carrying out service $j \in S$ after its soft time window.

20 min) will be discounted. Then, the total worked time will be 7 h and 20 min. In the schedule of Fig. 1(b), even though it has two breaks longer than 2 h, only the second one (with a duration of 3 h and 50 min) will not be paid. This means that the working time will be 8 h and 50 min. Therefore, Fig. 1 reveals the importance of correctly scheduling the largest break of a route.

According to the information provided by the company, there are two clearly differentiated objectives to be taken into account when designing the schedules: the cost associated with the schedules and the welfare. The cost of the schedule represents the expenses associated with the caregivers carrying out their routes. These expenses account for the working time and the non-contractual overtime. The welfare of the users represents the degree of well being and satisfaction that users present according to the schedules, and it combines two elements. On one hand, the affinity of the users with the assigned caregivers should be maximized and, on the other one, the penalization of violating the soft time windows should be minimized.

Since the company prefers to sacrifice cost savings for the sake of having a stable schedule that remains fairly unchanged over time, the most important objective will be the welfare.

A detailed example to illustrate the problem tackled by the company is presented in Appendix.

2.1. The MILP model

Next, the model that describes the HCSP under study is presented (see Table 2).

The formulation of the problem is:

$$f_1 = \min \omega_1 \sum_{i\in N}\sum_{d\in D}\sum_{j\in S}\sum_{k\in S^1} \lambda^i_j x^{id}_{jk} + \omega_2 \sum_{j\in S}(v^{start}_j + v^{end}_j) \tag{1}$$

$$f_2 = \min \omega_3 \sum_{i\in N} z^i + \omega_4 \sum_{i\in N}\sum_{d\in D}(t^{id}_s - t^{id}_0 - \hat{r}^{id}) \tag{2}$$

Subject to

$$\sum_{i\in N}\sum_{d\in D}\sum_{k\in S^1_{-j}} x^{id}_{jk} = 1 \qquad \forall j \in S \tag{3}$$

$$\sum_{i\in N}\sum_{d\in D}\sum_{j\in S^0_{-k}} x^{id}_{jk} = 1 \qquad \forall k \in S \tag{4}$$

$$\sum_{d\in D}\sum_{k\in S^1_{-j}} x^{id}_{jk} \le \rho^i_j \qquad \forall i \in N, \forall j \in S \tag{5}$$

$$\sum_{k\in S^1} x^{id}_{0k} = 1 \qquad \forall i \in N, \forall d \in D \tag{6}$$

$$\sum_{j\in S^0} x^{id}_{js} = 1 \qquad \forall i \in N, \forall d \in D \tag{7}$$

$$\sum_{j\in S^0_{-h}} x^{id}_{jh} - \sum_{k\in S^1_{-h}} x^{id}_{hk} = 0 \qquad \forall i \in N, \forall d \in D, \forall h \in S \tag{8}$$

$$\underline{\alpha}^d_j \sum_{k\in S^1_{-j}} x^{id}_{jk} \le t^{id}_j \le (\bar{\alpha}^d_j - \eta_j) \sum_{k\in S^1_{-j}} x^{id}_{jk} \qquad \forall i \in N, \forall d \in D, \forall j \in S \tag{9}$$

$$t_j^{id} + (\eta_j + \theta_{jk})x_{jk}^{id} \le t_k^{id} + \bar{\alpha}_j^d(1 - x_{jk}^{id}) \quad \forall i \in N, \forall d \in D, \forall j \in S, \forall k \in S^1, j \ne k \tag{10}$$

$$t_0^{id} \ge \underline{\gamma}^{id} \quad \forall i \in N, \forall d \in D \tag{11}$$

$$t_s^{id} \le \bar{\gamma}^{id} \quad \forall i \in N, \forall d \in D \tag{12}$$

$$t_0^{id} \le t_k^{id} + \bar{\gamma}^{id}(1 - x_{0k}^{id}) \quad \forall i \in N, \forall d \in D, \forall k \in S^1 \tag{13}$$

$$t_0^{id} \ge t_k^{id} - \bar{\gamma}^{id}(1 - x_{0k}^{id}) \quad \forall i \in N, \forall d \in D, \forall k \in S^1 \tag{14}$$

$$t_s^{id} \le (t_j^{id} + \eta_j) + \bar{\gamma}^{id}(1 - x_{js}^{id}) \quad \forall i \in N, \forall d \in D, \forall j \in S \tag{15}$$

$$t_s^{id} - t_0^{id} - \hat{r}^{id} \le v^{id} \quad \forall i \in N, \forall d \in D \tag{16}$$

$$z^i \ge \sum_{d \in D} \left(t_s^{id} - t_0^{id} - \hat{r}^{id}\right) - v^i \quad \forall i \in N \tag{17}$$

$$r^{id} \ge t_k^{id} - (t_j^{id} + \eta_j + \theta_{jk}) - \bar{\gamma}^{id}(1 - x_{jk}^{id}) \quad \forall i \in N, \forall d \in D, \forall j \in S, \forall k \in S, j \ne k \tag{18}$$

$$r^{id} \le t_k^{id} - (t_j^{id} + \eta_j + \theta_{jk}) + \bar{\gamma}^{id}(1 - x_{jk}^{id}) + \bar{\gamma}^{id}(1 - y_{jk}^{id}) \quad \forall i \in N, \forall d \in D, \forall j \in S, \forall k \in S, j \ne k \tag{19}$$

$$r^{id} \le \bar{\gamma}^{id}(1 - \bar{y}^{id}) \quad \forall i \in N, \forall d \in D \tag{20}$$

$$\sum_{j \in S} \sum_{k \in S_{-j}} y_{jk}^{id} + \bar{y}^{id} = 1 \quad \forall i \in N, \forall d \in D \tag{21}$$

$$y_{jk}^{id} \le x_{jk}^{id} \quad \forall i \in N, \forall d \in D, \forall j \in S, \forall k \in S, j \ne k \tag{22}$$

$$r^{id} - \pi_{min} \ge \pi_{min}(u^{id} - 1) \quad \forall i \in N, \forall d \in D \tag{23}$$

$$r^{id} - \pi_{min} + \varepsilon \le (\bar{\gamma}^{id} - \underline{\gamma}^{id})u^{id} \quad \forall i \in N, \forall d \in D \tag{24}$$

$$\hat{r}^{id} \le (\bar{\gamma}^{id} - \underline{\gamma}^{id})u^{id} \quad \forall i \in N, \forall d \in D \tag{25}$$

$$\hat{r}^{id} \le r^{id} \quad \forall i \in N, \forall d \in D \tag{26}$$

$$\hat{r}^{id} \ge r^{id} - (\bar{\gamma}^{id} - \underline{\gamma}^{id})(1 - u^{id}) \quad \forall i \in N, \forall d \in D \tag{27}$$

$$v_j^{start} \ge \sum_{d \in D} \left(\underline{\beta}_j^d \sum_{i \in N} \sum_{k \in S_{-j}^1} x_{jk}^{id} - \sum_{i \in N} t_j^{id} \right) \quad \forall j \in S \tag{28}$$

$$v_j^{end} \ge \sum_{d \in D} \left(\sum_{i \in N} t_j^{id} + (\eta_j - \bar{\beta}_j^d) \sum_{i \in N} \sum_{k \in S_{-j}^1} x_{jk}^{id} \right) \quad \forall j \in S \tag{29}$$

$$x_{jk}^{id} \in \{0,1\} \quad \forall i \in N, \forall d \in D, \forall j \in S^0, \forall k \in S^1, j \ne k \tag{30}$$

$$t_j^{id} \ge 0 \quad \forall i \in N, \forall j \in S^{01}, \forall d \in D \tag{31}$$

$$y_{jk}^{id} \in \{0,1\} \quad \forall i \in N, \forall d \in D, \forall j \in S, \forall k \in S, j \ne k \tag{32}$$

$$\bar{y}^{id}, u^{id} \in \{0,1\};\ r^{id}, \hat{r}^{id} \ge 0 \quad \forall i \in N, \forall d \in D \tag{33}$$

$$z^i \ge 0 \quad \forall i \in N \tag{34}$$

$$v_j^{start}, v_j^{end} \ge 0 \quad \forall j \in S \tag{35}$$

The objective function considers four different elements, which can be divided into two groups:

- Objective (1) accounts for the welfare of users, i.e., the total affinity between services and caregivers and the penalization of performing services outside their soft time windows. In order to set the weights ω_1 and ω_2, we take into account the fact that $v_j^{start} \le \max_{d \in D}\{\underline{\beta}_j^d - \underline{\alpha}_j^d\}$ and $v_j^{end} \le \max_{d \in D}\{\bar{\alpha}_j^d - \bar{\beta}_j^d\}$ for all $j \in S$, which means that setting $\omega_1 = -\max\{1, \sum_{j \in S}[\max_{d \in D}\{\underline{\beta}_j^d - \underline{\alpha}_j^d\} + \max_{d \in D}\{\bar{\alpha}_j^d - \bar{\beta}_j^d\}]\}$ and $\omega_2 = 1$ guarantees that the affinity will be prioritized over the soft time windows.[1]
- Objective (2) refers to schedule cost, which is composed by the amount of overtime of the caregivers and their total worked time. Since the units coincide, we set $\omega_3 = \omega_4 = 1$.

Concerning the constraints, (3)–(5) ensure that all services must be carried out. Constraints (6)–(8) determine the routes of the caregivers and (9)–(15) how the services must be scheduled according to the hard time windows. The working time of the caregivers is obtained in Constraints (16)–(17). Constraints (18)–(27) assess the largest daily break of the caregivers. The deviation from the soft time windows is calculated in Constraints (28)–(29). Lastly, Constraints (30)–(35) establish the domain of the variables.

3. Heuristic algorithm

The MILP formulation can only be solved in small instances with a state of the art optimization solver because of the complexity of the problem. For this reason, we designed a tailored algorithm, based on the Adaptive Large Neighbourhood Search (ALNS) method proposed by Ropke and Pisinger (2006). This methodology has been proved to be more successful at solving different types of vehicle routing problems than other traditional methods (for more information, see Pisinger and Ropke (2007)).

Following the general scheme of an ALNS method, the developed algorithm first generates a feasible initial solution and then introduces movements to modify the routes, that is, it changes the order of the services and/or the caregivers assigned to the route. Once the routes are established, to correctly evaluate the objective function, we need to obtain the starting times of each service. To this aim, we developed a heuristic method, which is the most innovative part of our framework and plays a key role in the good performance of the algorithm.

[1] Note that we consider $\omega_1 < 0$ because we want to maximize the affinity levels in a minimization problem.

Table 3
Removal operators.

Operator	Description
Random removal	The services to remove from the route are selected at random.
Related removal	This operator iteratively removes from the route the service that is most related to one of the already removed services, which is chosen at random. The first service to remove is randomly selected and the level of relation between two services is measured by the likeness of their time windows and the day they belong to.
Cost removal	The services that contribute the most to the objective function value are iteratively removed from the routes.
1-Route removal	It removes complete routes, selected at random, from the solution until the required number of services have been deleted.
2-Route removal	It removes two complete routes, selected at random, from the solution.

Table 4
Insertion operators.

Operator	Description
Basic Greedy (BG)	It adds to the schedule the service that, when inserted at its best position, results in the least objective function increase.
Random Greedy (RG)	The services are iteratively chosen at random and, then, are scheduled at the best position according to the objective function value[a].
Different Caregiver BG	Similar to the basic greedy but, in this case, the services have to be assigned to a different caregiver than the one who was attending them before the destruction phase.
Different Caregiver RG	As the random greedy but, in this case, the services have to be assigned to a different caregiver than the one who was attending them before the destruction phase.

[a]This operator is also used to generate the initial feasible solution.

Algorithm 1 describes the pseudocode of the ALNS method, which is based on a destroy and repair methodology that considers multiple operators to remove (insert) services from (to) the solutions.

The method starts with a feasible solution. Then, the operators are randomly chosen according to probabilities that are updated after each iteration, to use more frequently the ones providing better solutions. After obtaining the new solution, the current one is updated using the simulated annealing acceptance criterion. This criteria accepts the new solution with probability $exp(-(cost(x^*) - cost(x))/T_i)$, where T_i is the temperature that decreases after each iteration i according to a cooling parameter $0 < \beta < 1$, the initial temperature $T_0 > 0$ and the formula $T_i = \beta^i T_0$.

Algorithm 1: Adaptive Large Neighbourhood Search

Data: Initial solution x, Removal operators Ω_{rem}, Insertion operators Ω_{ins}

$x' \leftarrow x$, $\omega_{rem} \leftarrow (1, ..., 1)$, $\omega_{ins} \leftarrow (1, ..., 1)$
while *stopping criteria not met* **do**
  $removalOperator \leftarrow$ ***chooseRandom***$(\omega_{rem}, \Omega_{rem})$
  $insertionOperator \leftarrow$ ***chooseRandom***$(\omega_{ins}, \Omega_{ins})$
  $\hat{x} \leftarrow$ ***removalOperator***(x)
  $x^* \leftarrow$ ***insertionOperator***$(\hat{x})$ $\Rightarrow$ *Call* ***Algorithm 2***
  if $f(x^*) < f(x')$ **then**
    $x' \leftarrow x^*$
  $x \leftarrow$ ***acceptanceCriteria***(x^*, x')
  $\omega_{rem} \leftarrow$ ***updateWeights***(ω_{rem}), $\omega_{ins} \leftarrow$ ***updateWeights***(ω_{ins})
return x'

Although we have outlined the general structure of the ALNS algorithm, it is still necessary to describe the specific features relevant to our problem: the methodology used to obtain the initial solution, the insertion and removal operators and, finally, how to evaluate the objective function.

The removal operators are used to destroy part of the solution, depending on a parameter that determines the proportion of services that we allow to be removed from the schedule during each iteration. The five removal operators are described in Table 3.

The insertion operators are used to obtain the initial schedule and to reconstruct the solution destroyed by the removal operators. The four insertion operators are described in Table 4.

Since we deal with a routing and scheduling problem, to obtain the schedule for the evaluation of the insertion operators it is not only necessary to know the routes of each caregiver, but also the starting times of every service. Therefore, to obtain the services schedules, we developed one approach (Algorithm 2) to evaluate the objective function in a lexicographical order.

3.1. Approach to evaluate the objective function

Fig. 2 describes Algorithm 2 designed to obtain the schedule of a route that first optimizes the welfare of the users and, after that, the cost of the schedule.

The implementation scheme of the approach can be divided into three parts, explained in the following sections.

3.1.1. Starting times

This first part of the approach, described in Algorithm 2.1, obtains the earliest and latest starting times for hard (lines 1–10) and soft (lines 11–14) time windows. After that, the route is divided into blocks, where two services (and the ones between them) belong to the same block if their soft time windows overlap (line 15).

Example 3.1. In Fig. 3, an example[2] of a route with six services is presented. For simplicity, all the services have equal duration and there is no travel time between each pair of services.

Fig. 4 shows the earliest, t_j^e, and latest, t_j^l, starting times according to hard time windows (lines 1–10); and to soft time windows (lines 11–14), $\underline{b}_j$ and $\bar{b}_j$. The route can be divided into three blocks with overlapping soft time windows: $\Delta = \{\{1\}, \{2\}, \{3, 4, 5, 6\}\}$ (line 15).

[2] The example will be used to illustrate how the whole approach to schedule a route works.

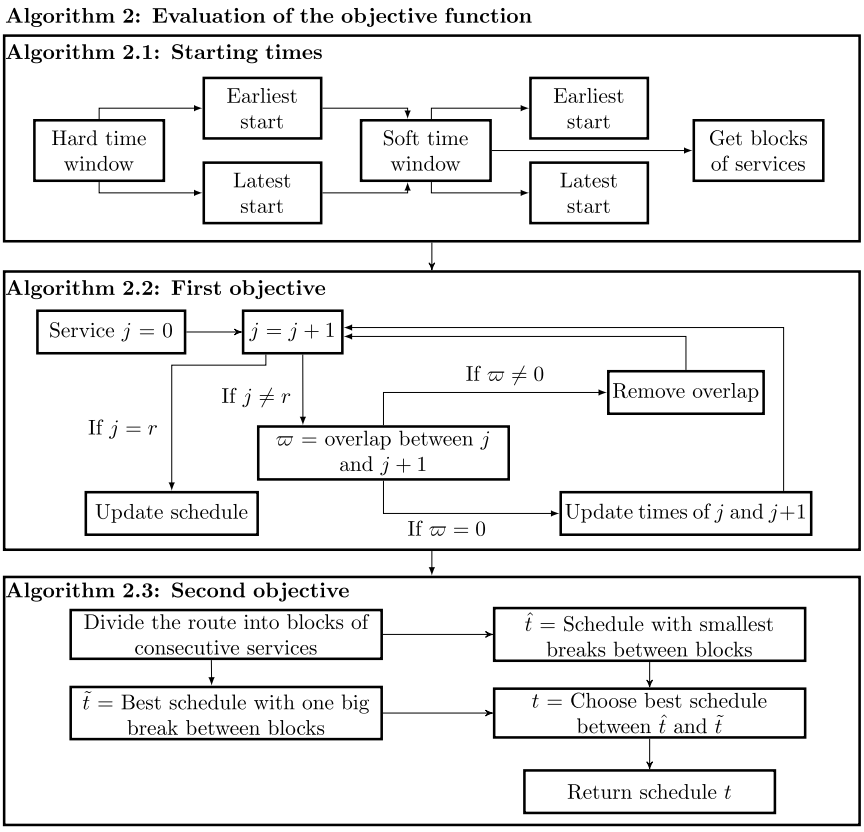


Fig. 2. Scheme to obtain the schedule of a route ($R = [1, \dots, r]$).

Algorithm 2.1: Obtain earliest and latest starting times, divide the route into blocks

1 **for** $j \in R$ **do** // Get earliest start for the services, according to hard time windows
2 | **if** $j = 1$ **then**
3 | | $t^e_j \leftarrow \max\{\underline{\alpha}^d_j, \underline{\gamma}^{id}\}$
4 | **else**
5 | | $t^e_j \leftarrow \max\{\underline{\alpha}^d_j, t^e_{j-1} + \eta_j + \theta_{j-1,j}\}$
6 **for** $j \in R$ **do** // Get latest start for the service, according to hard time windows
7 | **if** $j = r$ **then**
8 | | $t^l_j \leftarrow \min\{\bar{\alpha}^d_j - \eta_j, \bar{\gamma}^{id} - \eta_j\}$
9 | **else**
10 | | $t^l_j \leftarrow \min\{\bar{\alpha}^d_j - \eta_j, t^l_{j+1} - \theta_{j,j+1} - \eta_j\}$
11 **for** $j \in R$ **do** // Get earliest start for the services, according to soft time windows
12 | $\underline{b}_j \leftarrow \min\{\max\{\underline{\beta}^d_j, t^e_j\}, t^l_j\}$
13 **for** $j \in R$ **do** // Get latest start for the services, according to soft time windows
14 | $\bar{b}_j \leftarrow \max\{\min\{\bar{\beta}^d_j - \eta_j, t^l_j\}, t^e_j\}$
15 $\Delta \leftarrow$ ***getBlocksSTW*** (R) // Get blocks of services with overlapping soft time windows

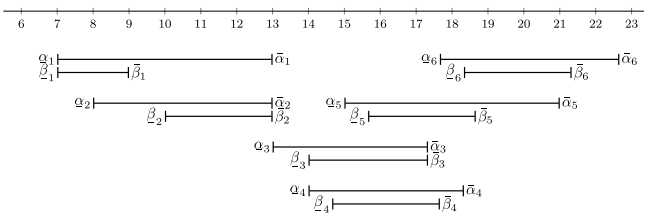


Fig. 3. Route to schedule.

3.1.2. First objective

The goal of this part of the approach is to get the schedule with the best welfare, and it is described in Algorithm 2.2. When setting the position of the services, it is essential to know if there is an overlap between consecutive services (line 4). Thus, we can distinguish two cases:

- If there is no overlap, the starting time of the first service is updated. This could modify the earliest starting time of the soft time window for the second service (lines 32 and 35).
- Otherwise, the blocks with consecutive services are joined forming a single block (line 6), which is scheduled at its earliest possible time (lines 7–12). To obtain the best starting time of the

Algorithm 2.2: Obtain the schedule with best welfare value

1 $\bar{R} \leftarrow \emptyset$ // Schedule services to get the best welfare value
2 **for** $j \in R$ **do**
3 **if** $j \neq r$ *and* $j \notin \bar{R}$ **then**
4 $\varpi \leftarrow \underline{b}_j + \eta_j + \theta_{j,j+1} - \bar{b}_{j+1}$ // Get overlap with follower
5 **if** $\varpi > 0$ **then**
6 $\bar{\delta} \leftarrow$ ***getBlockOfServices***$(\Delta, j, j+1)$ // Join the blocks that contain the services
7 **if** $\bar{\delta}_1 = 1$ **then** // Schedule the block $\bar{\delta} = [\bar{\delta}_1, ..., \bar{\delta}_b]$ at its earliest time
8 $t_{\bar{\delta}_r} \leftarrow t^e_{\bar{\delta}_r}$
9 $t_k \leftarrow \min\{\bar{\alpha}^d_k - \eta_k, t_{k+1} - \theta_{k,k+1} - \eta_k\}\ \forall k \in [\bar{\delta}_{b-1}, ..., \bar{\delta}_1]$
10 **else**
11 $t_k \leftarrow \max\{\underline{\alpha}^d_k, t^{id}_{k-1} + \eta_{k-1} + \theta_{k-1,k}\}\ \forall k \in [\bar{\delta}_1, ..., \bar{\delta}_b]$
12 $t_k \leftarrow \min\{\bar{\alpha}^d_k - \eta_k, t_{k+1} - \theta_{k,k+1} - \eta_k\}\ \forall k \in [\bar{\delta}_{b-1}, ..., \bar{\delta}_1]$
13 $delay \leftarrow True$ // Delay the block to improve welfare
14 **while** $delay = True$ **do**
15 $E, improve \leftarrow$ ***getPossibleDelays***$(\bar{\delta}, t_{\bar{\delta}}, t^l)$ // Get possible delay times
16 **if** $improve$ *is* $False$ *and* $\bar{\delta}_1 \neq 1$ **then** // Check if the welfare improves
17 $\bar{\delta} \leftarrow previousblock + \bar{\delta}$ // Add the previous block
18 $\{t_k\}_{k \in \bar{\delta}} \leftarrow$ ***getBlockSchedule***$(\bar{\delta})$ // Schedule the block at its earliest time
19 **else if** $improve$ *is* $True$ **then**
20 $\epsilon_{max} \leftarrow \underline{\beta}^d_{\hat{\delta}_1} - (t_{\bar{\delta}_b} + \eta_{\bar{\delta}_b} + \theta_{\bar{\delta}_b,\hat{\delta}_1})$ // Get delay time needed to reach the follower block $\hat{\delta} = [\hat{\delta}_1, ..., \hat{\delta}_b]$
21 **if** $\bar{\delta}$ *is not the last block and* $\epsilon_{max} < \max\{E\}$ **then**
22 $E \leftarrow \{\epsilon_i \in E : \epsilon_i < \epsilon_{max}\} \cup \{\epsilon_{max}\}$
23 $t_{\bar{\delta}} \leftarrow$ ***delayBlock***$(\bar{\delta})$ // Delay the block the maximum amount of time that reduces welfare
24 **if** $\bar{\delta}$ *is not the last block and* $\epsilon_{final} = \epsilon_{max}$ **then**
25 $\bar{\delta} \leftarrow \bar{\delta} + \hat{\delta}$ // Add following block if necessary
26 $t_l \leftarrow \max\{\underline{\alpha}^d_l, t_{l-1} + \eta_{l-1} + \theta_{l-1,l}\}\ \forall l \in [\hat{\delta}_1, ..., \hat{\delta}_b]$ // Set times for the block
27 **else**
28 $delay \leftarrow False$
29 **else**
30 $delay \leftarrow False$
31 $\bar{R} \leftarrow \bar{R} + \bar{\delta}$
32 **else**
33 $t_j \leftarrow \underline{b}_j$, $\bar{R} \leftarrow \bar{R} + j$, $\underline{b}_{j+1} \leftarrow \max\{t_j + \eta_j + \theta_{j,j+1}, \underline{b}_{j+1}\}$ // Update elements
34 **else if** $j = r$ *and* $j \notin \bar{R}$ **then**
35 $t_j \leftarrow \underline{b}_j$

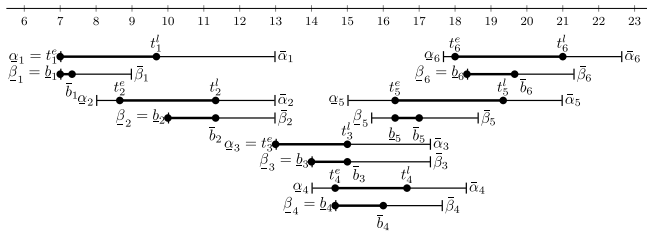


Fig. 4. Earliest and latest starting times, according to hard and soft time windows.

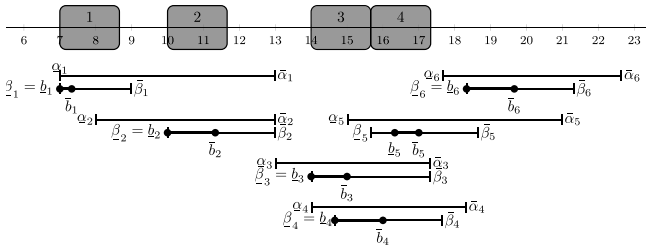


Fig. 5. Schedule for services 1, 2, 3 and 4.

new block, we will take into account that each service in the block could be positioned at the beginning or at the end of its soft time window (line 15). This procedure of joining blocks continues until there is no overlap between consecutive blocks (lines 16–28).

Example 3.2. Considering Example 3.1, a feasible schedule can be obtained by setting the following starting times: service 1 at $\underline{b}_1$, service 2 at $\underline{b}_2$, service 3 at $\underline{b}_3$ and service 4 at $\underline{b}_3 + \eta_3$ (lines 3 and 33). As it can be seen in Fig. 5, there is no soft time windows penalization with this schedule.

Now, let us assume that service 5 is added. Fig. 6 shows that if this service is scheduled at its latest time, according to its soft time window, then there is an overlap with service 4 (line 4).

In Fig. 7, the block composed of services 3, 4, 5 and 6 is handled as if it were a single service and it is scheduled at its earliest time (lines 6–12). The dotted line indicates the amount of time outside the soft time windows of the services.

Fig. 8 presents three ways of obtaining delay times for the block (lines 15, 19–28). For this example the best move is to delay the block until service 5 reaches the end of its soft time window (Fig. 8(b)), because a bigger delay would not provide better solutions (Fig. 8(c)).

Algorithm 2.3: Obtain the schedule with best cost value

1 $\Delta \leftarrow$ ***getBlocksConsecutiveServices***(R, t) // Get block of consecutive services
2 **for** $\delta \in \Delta$ **do** // Get time window for each block so the welfare is maintained
3 **if** δ *only has one service,* j **then**
4 $\underline{a}_j \leftarrow \underline{\beta}^d_j$, $\bar{a}_j \leftarrow \bar{\beta}^d_j - \eta_j$
5 **else**
6 $\lambda_a \leftarrow \min_{k\in\delta}\{t^{id}_k - t^e_k\}$, $\lambda_d \leftarrow \min_{k\in\delta}\{t^l_k - t^{id}_k\}$ // Get maximum advance and delay
7 $E_d, E_a \leftarrow$ ***getDelayAdvanceTimes***$(\delta, t_\delta, \lambda_a, \lambda_d)$ // Get possible delay and advance times
8 $\epsilon_d \leftarrow$ ***getDelayTime***(E_d, t_δ) // Get maximum delay time that improves or maintains the welfare
9 $\epsilon_a \leftarrow$ ***getAdvanceTime***(E_a, t_δ) // Get maximum advance time that improves or maintains the welfare
10 **for** $j \in \delta$ **do** // Get the time window for the services
11 $\underline{a}_j \leftarrow t_j - \epsilon_a$, $\bar{a}_j \leftarrow t_j + \epsilon_d$
12 $\underline{a}, \bar{a} \leftarrow$ ***updateTimeWindow***$(\Delta, a, \bar{a})$ // Update the time window so the block can be scheduled at $\underline{a}$ and $\bar{a}$
13 **if** *there is only one block* **then**
14 $\bar{t} \leftarrow t$
15 **else**
16 $j \leftarrow$ *first service of second block*
17 $\hat{t}_l \leftarrow \bar{a}_l \ \forall l \in first\ block$ // Make all breaks between blocks as small as possible
18 $\hat{t}_{jl} \leftarrow \max\{\underline{a}_l, t_{l-1} + \eta_{l-1} + \theta_{l-1,l}\} \ \forall l \in [j, .., r]$
19 **while** $\delta \in \Delta$ *is not the last one* **do** // Make break after δ as big as possible
20 $j \leftarrow$ *first service of* δ, $k \leftarrow$ *last service of* $\delta + 1 = \hat{\delta}$
21 $\bar{t}_l \leftarrow \underline{a}_l \ \forall l \in [\delta_1, ..., \delta_b]$, $\bar{t}_l \leftarrow \bar{a}_l \ \forall l \in [\hat{\delta}_1, ..., \hat{\delta}_b]$
22 $\bar{t}_l \leftarrow \min\{\bar{a}_l - \eta_l, \bar{t}_{l+1} - \theta_{l,l+1} - \eta_l\} \ \forall l \in [j-1, ..., 1]$
23 $\bar{t}_l \leftarrow \max\{\underline{a}_l, \bar{t}_{l-1} + \eta_{l-1} + \theta_{l-1,l}\} \ \forall l \in [k+1, ..., r]$
24 $\tilde{t} \leftarrow$ ***chooseBestCostSchedules***$(\bar{t}, \tilde{t})$
25 $t \leftarrow$ ***chooseBestCostSchedules***$(\hat{t}, \tilde{t})$ // Choose the schedule with best cost value
26 **return** t

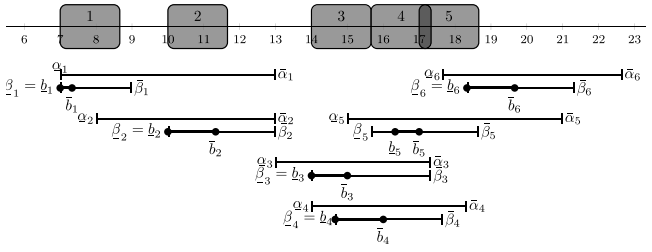


Fig. 6. Schedule adding service 5.

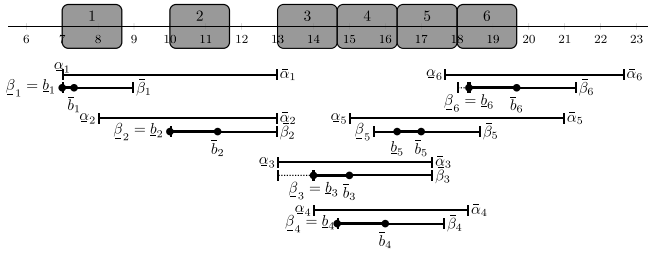


Fig. 7. Schedule with a block of services.

3.1.3. Second objective

To improve the cost of the schedule, the approach (Algorithm 2.3) divides the route into blocks of services without breaks (line 1). Then, for each block, we compute a new time window guaranteeing that the value of the second objective remains unchanged (lines 2–12). Within this time window, it may be possible to reduce the cost. To this aim, two different strategies are proposed:

- We can make all the breaks between blocks as small as possible (lines 16–18).
- We can make one of the breaks as big as possible and reduce the other ones (lines 19–24).

The option resulting in the minimum cost is selected (line 25).

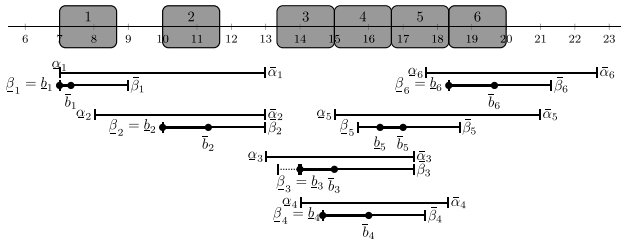
Example 3.3. Once a solution with the best penalization is obtained in Example 3.2, it is time to explore the second objective. To this aim, the route is divided in blocks of consecutive services: $\Delta = \{\{1\}, \{2\}, \{3, 4, 5, 6\}\}$ (line 1). As shown in Fig. 9, the earliest ($\underline{a}_j$) and latest ($\bar{a}_j$) starting times for the services, guaranteeing that the penalization of the blocks will not change, are computed (lines 2–12).

To get the schedule with the minimum cost, two options might be considered: reduce the breaks between services (see Fig. 10(a), lines 16–18) or make one of the breaks as big as possible (see Figs. 10(b) and 10(c), lines 19–24). In this particular case, Fig. 10(b) presents the optimal solution, since increasing the break between services 1 and 2 results in the best cost (line 25).

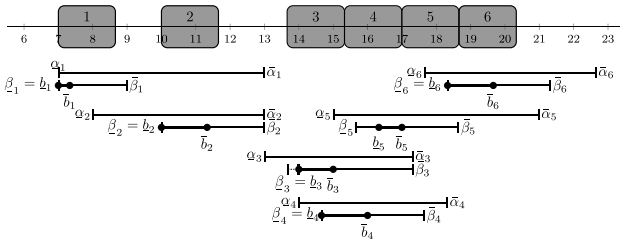
4. Computational experiments

In this section we present the computational study carried out in order to check the behaviour of the heuristic algorithm previously described.

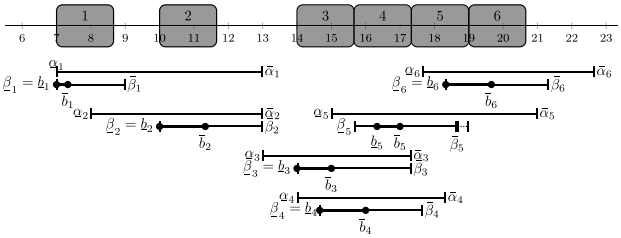
The algorithm has been implemented in Python 3.7 (Van Rossum and Drake, 2009), and the MILP problem has been solved with Gurobi 9.1.1 (Gurobi Optimization, 2021) via its Python interface. All the experiments were run in a machine Intel(R) Xeon(R) Gold 6146 CPU 3.20 GHz, with 16 GB of RAM, 2 cores and 100 GB of hard drive, located in Centre for Information and Communications Technology Research (CITIC).



(a) Service 6 starts at its soft time window.



(b) Service 5 ends at its soft time window.



(c) Service 3 starts at its soft time window.

Fig. 8. Analysing delays of the block $\{3, 4, 5, 6\}$.

Table 5
Data instances.

S	N	Duration		Length of htw		Length of stw		Travel time		Size
		min	max	min	max	min	max	min	max	
10	2	10	90	65	3380	64	3367	1	75	496 × 950
15	2	10	90	83	1343	75	1258	1	89	1036 × 2010
25	4	10	90	53	896	12	93	1	84	5382 × 10560
50	6	10	90	49	1226	11	93	1	96	31048 × 61490
100	14	10	90	20	1225	12	93	1	101	284512 × 566210

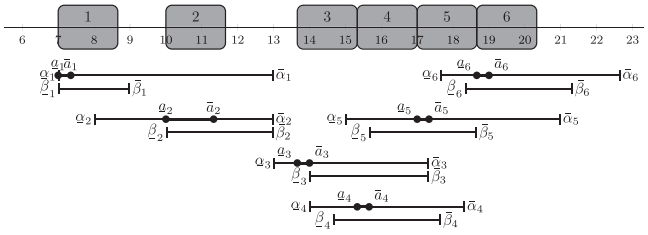


Fig. 9. Earliest and latest starting times of the services.

4.1. Study of the parameters of the algorithm

The goal of this analysis is not specifically devoted to compare the MILP model with our method, but mainly to allow us to adjust the parameters of the algorithm using data that captures diverse scenarios. Thus, the experiments are based on the data from Solomon (1987), which provide a great diversity in the location of the services. These instances represent 100-customer euclidean problems, with the following characteristics: services to carry out, locations, duration, hard time windows, caregivers availability and total working times. Further, some additional data required by our particular problem were randomly generated: services–caregivers affinity levels and soft time windows of the services.

Table 5 contains a basic description of 5 types of Solomon instances, in terms of the number of services (S) and the number of caregivers (N). For each type, 10 different instances were randomly generated, specifying: minimum and maximum duration of the services, minimum and maximum length of the hard time windows (htw) and soft time windows (stw), minimum and maximum travel times between the services and size of the problems (number of variables × number of constraints). In addition, the affinity levels were randomly set to 3 or 5.

The MILP has been solved by using Gurobi with a time limit of 12 h. The results are presented in Table 6. Note that although Gurobi finds a feasible solution for 80% of the instances, it only ensures the global optimality in 45% of them.

Regarding the configuration of the heuristic algorithm, the parameters considered are the number of iterations (50, 100, 150, 200, 250, 500, 750 and 1000) and the proportion of solution to be destroyed, from now on p, (25%, 50%, 75% and 100%). Further, an automatic version that dynamically updates this parameter is employed, denoted as *auto_p*, which is a method that starts with a proportion of p and decreases after each iteration of the algorithm (*auto*_25%, *auto*_50%, *auto*_75% and *auto*_100%). Each combination of parameters has been run 5 times with a time limit of 1 h.

A detailed comparison of the solutions given by the algorithm and the ones found using Gurobi, as well as an in-depth study of the parameter p, are presented in Appendix B. In summary, for small instances (10 and 15 services), it can be seen that the algorithm behaves better than Gurobi for the biggest values of p. Furthermore, the results evidence that as far as the size of the instances increases, the better solutions are reached if the value of p is slightly reduced.

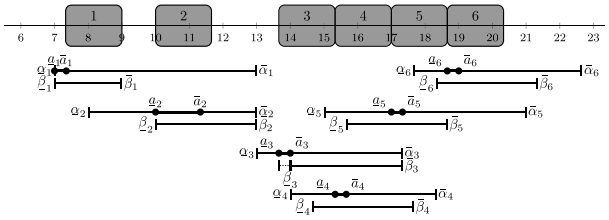
Figs. 11–13 present the evolution of the success rate (the proportion of times where the best solution is found) and the computational time

Table 6
Gurobi computational results.

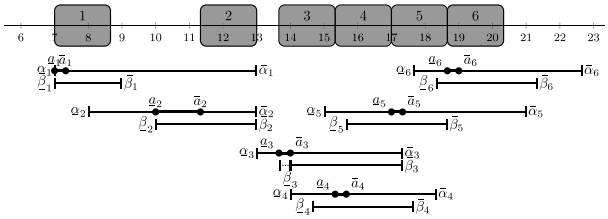
instance	feasible	opt	gap	secs	instance	feasible	opt	gap	secs
10_01	✓	✓	–	411.58	10_06	✓	✓	–	651.36
10_02	✓	✓	–	543.34	10_07	✓	✓	–	2346.73
10_03	✓	✓	–	1447.64	10_08	✓	✓	–	389.92
10_04	✓	?	1103.70	limit	10_09	✓	?	1160.16	limit
10_05	✓	✓	–	1681.82	10_10	✓	✓	–	558.92
15_01	✓	✓	–	4246.17	15_06	✓	?	1159.70	limit
15_02	✓	?	224.91	limit	15_07	✓	?	587.11	limit
15_03	✓	✓	–	5297.21	15_08	✓	?	1045.77	limit
15_04	✓	?	118.34	limit	15_09	✓	?	862.76	limit
15_05	✓	?	735.44	limit	15_10	✓	✓	–	41021.30
25_01	✓	✓	–	41.37	25_06	✓	✓	–	29.84
25_02	✓	✓	–	2009.30	25_07	✓	?	1.76	limit
25_03	✓	✓	–	37.02	25_08	–	–	–	–
25_04	✓	?	2.915	limit	25_09	✓	✓	–	28.26
25_05	–	–	–	–	25_10	✓	✓	–	158.94
50_01	–	–	–	–	50_06	✓	?	2.80	limit
50_02	✓	?	49.05	limit	50_07	–	–	–	–
50_03	✓	✓	–	410.92	50_08	✓	?	0.008	limit
50_04	✓	?	0.01	limit	50_09	✓	?	0.005	limit
50_05	✓	?	0.01	limit	50_10	✓	?	0.01	limit
100_01	✓	?	0.008	limit	100_06	–	–	–	–
100_02	–	–	–	–	100_07	–	–	–	–
100_03	✓	?	346.04	limit	100_08	–	–	–	–
100_04	–	–	–	–	100_09	–	–	–	–
100_05	✓	?	269.05	limit	100_10	✓	?	0.003	limit

Table 7
Real data instances.

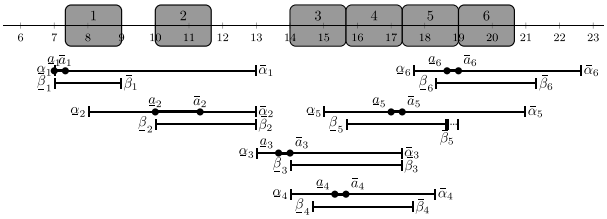
week	S	N	Duration		Length of htw		Length of stw		Travel time		Size (vars ×constr)
			min	max	min	max	min	max	min	max	
1	865	39	30	168	90	630	60	450	0	22	409242965 × 817811191
2	880	38	30	168	90	630	60	450	0	22	412686700 × 824704562
3	895	35	30	168	90	630	60	450	0	22	393163615 × 785700695
4	863	37	30	168	90	630	60	450	0	22	386464469 × 772290281
5	633	38	30	160	90	630	60	450	0	22	213674848 × 426868578
6	822	37	30	160	90	630	60	450	0	22	350646500 × 700684683
7	894	36	30	160	90	630	60	450	0	22	403494396 × 806345076
8	760	39	30	168	90	630	60	450	0	22	315995510 × 631398181
9	808	36	30	168	90	630	60	450	0	22	329657720 × 658733644
10	911	37	30	160	90	630	60	450	0	22	430610597 × 860547017
11	870	39	30	160	90	630	60	450	0	22	413983620 × 827288601
12	840	39	30	180	90	630	60	450	0	22	385949190 × 771243141
13	950	38	30	180	90	630	60	480	0	22	480891900 × 961061762
14	925	38	30	180	90	630	60	480	0	22	455934400 × 911165762
15	939	38	30	180	90	630	60	480	0	22	469828672 × 938943666



(a) Schedule with the smallest breaks.



(b) Schedule with the largest break between services 1 and 2.



(c) Schedule with the largest break between services 2 and 3.

Fig. 10. Procedure to improve the second objective.

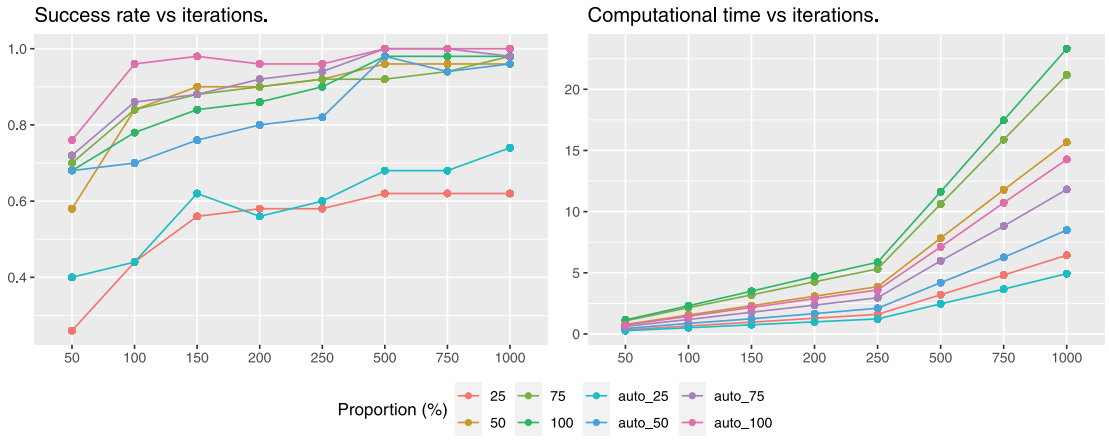


Fig. 11. Computational results for instances with 10 services.

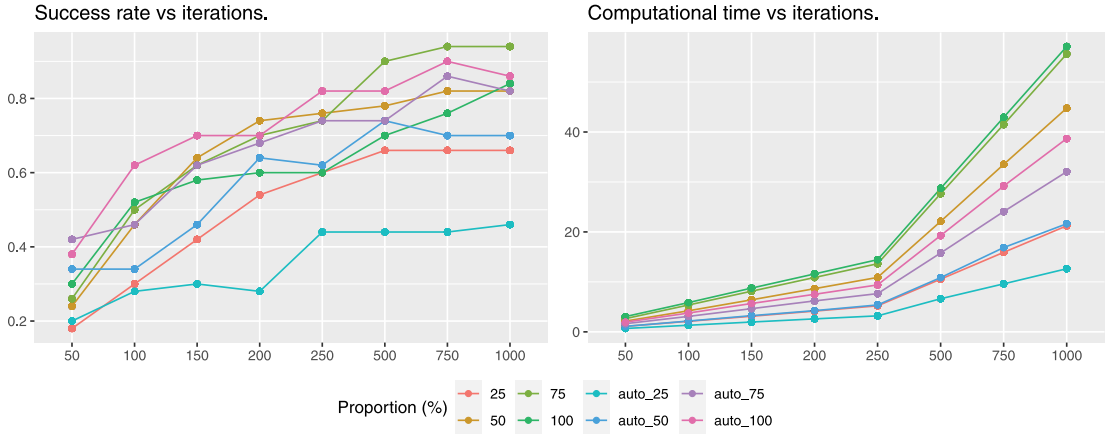


Fig. 12. Computational results for instances with 15 services.

(in seconds) along the iterations. Fig. 11 is related to instances with 10 services. Configurations $p \in \{auto\_75\%, auto\_100\%\}$ present really good behaviour according to the success rate and the computational time. It is clear that, for these instances, the worst configurations are $p \in \{25\%, auto\_25\%\}$. With regard to instances with 15 services, Fig. 12 shows that $p \in \{75\%, auto\_100\%\}$ are good choices, although $p = auto\_100\%$ is faster. Fig. 13 indicates that, for instances with 25 services, $p = auto\_75\%$ outperforms the other configurations of p in terms of success rate and computational time.

Figs. 14 and 15 evaluate mean RPD and computational time along the iterations for instances with 50 and 100 services. Notice that the Relative Percentage Deviation (RPD), of solution x with respect to solution x', is defined as $RPD(x) = \frac{f(x)-f(x')}{f(x')} \times 100$.

As far as instances with 50 services, $p = auto\_25\%$ is the best configuration in terms of mean RPD and computational time. For instances with 100 services only three configurations, $p \in \{25\%, auto\_25\%, auto\_50\%\}$, are able to solve the instances for 750 and 1000 iterations in the given time limit. From these configurations, the best one (according to the mean RPD and computational time) is $p = auto\_25\%$.

To conclude, as the number of services increases, it is desirable to reduce the initial value of the proportion in the automatic configurations. In this way, the lowest initial values for *auto* behave better in the largest instances, obtaining fast computational times.

4.2. Real data

Table 7 shows the data of 15 real instances, taken by the schedule of the company during consecutive weeks between years 2016 and 2017. For these instances, the number of services vary from 633 to 950 and their duration is between 30 and 180 min. The available caregivers for each week vary from 35 to 39 and the affinity levels between users and caregivers are 3, 4 or 5.

According to the results in the previous section, several experiments were run with small values of the automatic configuration of p, with p from 1 to 10. Further, it was considered as stopping criterion a time limit of 90 min. This analysis showed that $p = 1$ was a good choice, so it was selected to compare the algorithm with the company solution.

Fig. 16 shows the results obtained by the algorithm and the schedules used by the company during the considered weeks. Besides, Figs. 16(a), 16(c) and 16(e) are devoted to study the welfare, whereas Figs. 16(b), 16(d) and 16(f) are related to the cost.

It can be seen in Fig. 16(a) and 16(b) that the algorithm always finds better results than those employed by the company. In addition, if we break down the welfare into the affinity and the penalization, we also observe that the algorithm improves the results in both cases. Similar conclusions are obtained if we disaggregate the cost in overtime and worked time.

To explore the results in more detail, we will deeply analyse the solution with respect to the following characteristics: affinity, penalization, overtime, travel time, dynamic break and idle time.

Thus, Table 8 shows the results with respect to the affinity and the penalization on a weekly basis, both globally and on average per user. In all the weeks, the algorithm clearly outperforms the results of the company.

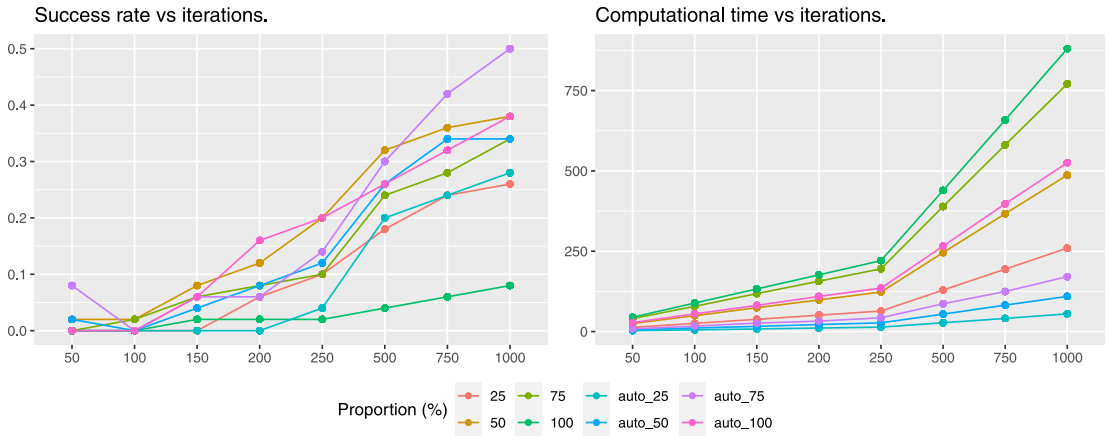


Fig. 13. Computational results for instances with 25 services.

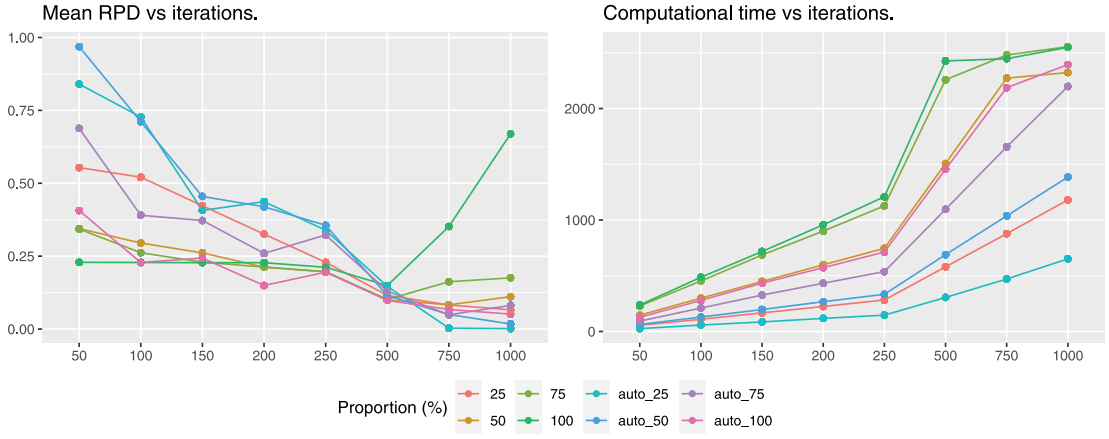


Fig. 14. Computational results for instances with 50 services.

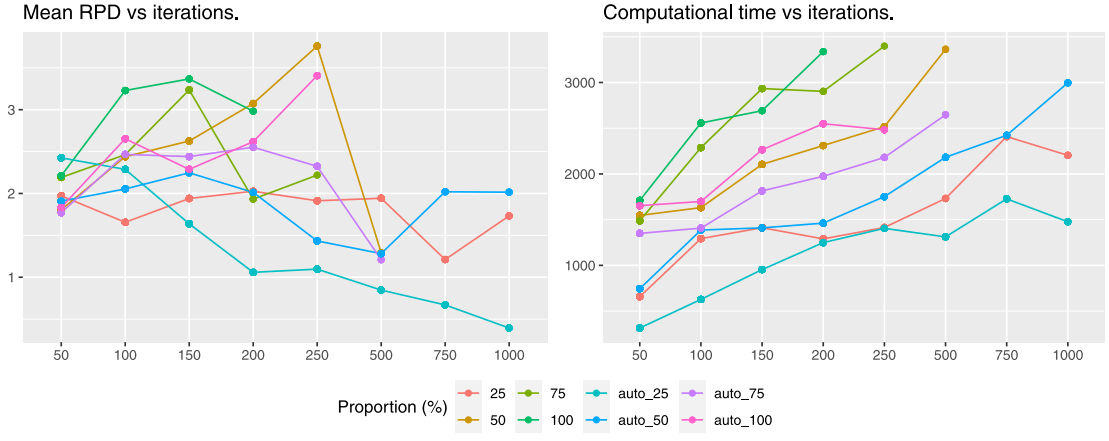
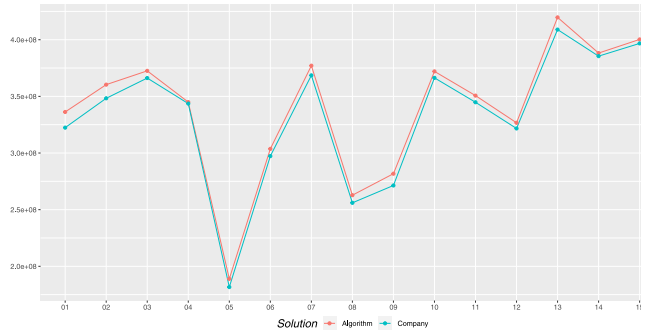


Fig. 15. Computational results for instances with 100 services.

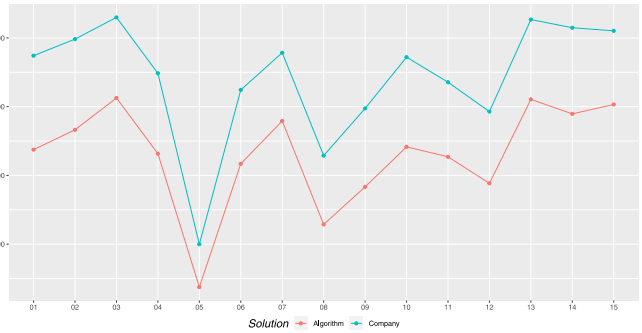
On the other hand, the quality of the solutions can be measured in terms of affinity obtaining the percentage of services where the best possible level of affinity is reached, which means that the best caregivers (according to the level of affinity) have been assigned to these services. Thus, Table 9 shows a comparison between the results of the company and the ones of the algorithm per week. In all the cases, the algorithm solution is clearly better than the solution proposed by the company. The percentages of services that have the highest affinity for the company and the algorithm are 81.04% and 88.87%, respectively, when considering all the services across all the weeks.

The soft time window penalization for the users and the services in all the weeks is represented in the boxplots of Figs. 17(a) and 17(b). The graphical results confirm a remarkable improvement with respect to the solutions of the company.

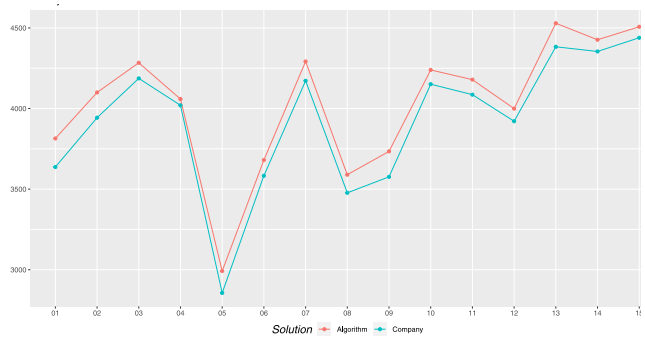
Table 10 shows the results per week with respect to the global overtime and worked time, as well as the average per caregiver with



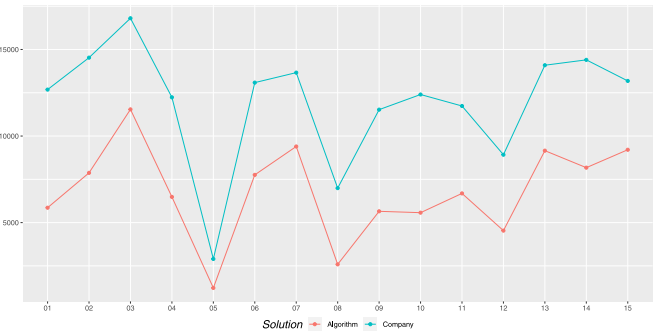
(a) Welfare for each week.



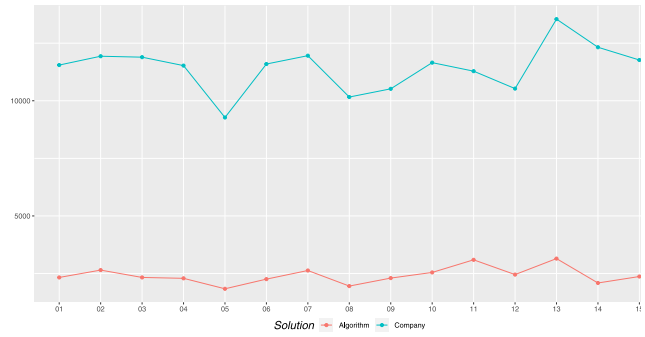
(b) Cost for each week.



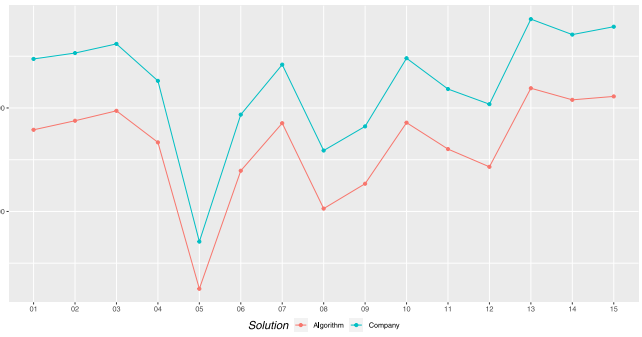
(c) Affinity for each week.



(d) Overtime for each week.

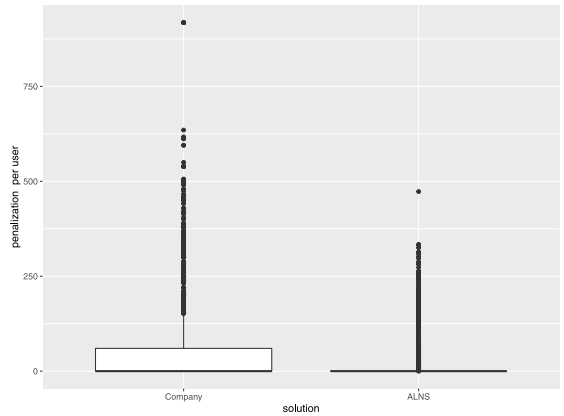


(e) Penalization for each week.

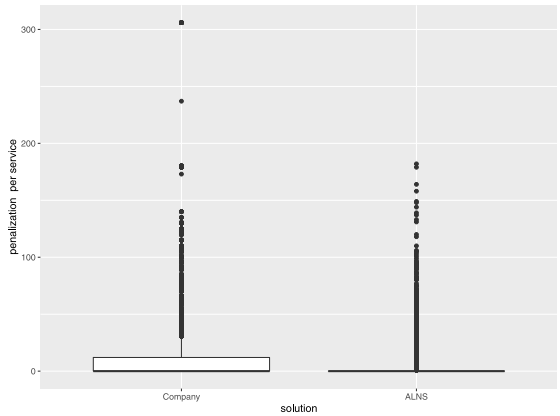


(f) Worked time for each week.

Fig. 16. Computational results for real instances.



(a) For the users.



(b) For the services.

Fig. 17. Soft time window penalization (all weeks).

Table 8
Computational results for real instances (welfare).

Week	Solution	Global			Per user	
		Welfare ($\times E8$)	Affinity ($\times E3$)	Penalization ($\times E3$)	Affinity	Penalization
1	Algorithm	3.36	3.81	2.33	4.33	2.73
	Company	3.22	3.64	11.55	4.14	13.73
2	Algorithm	3.60	4.10	2.65	4.66	2.76
	Company	3.48	3.94	11.93	4.50	14.94
3	Algorithm	3.72	4.28	2.33	4.76	2.75
	Company	3.66	4.19	11.89	4.66	14.66
4	Algorithm	3.44	4.06	2.29	4.66	3.18
	Company	3.44	4.02	11.53	4.62	15.19
5	Algorithm	1.88	2.99	1.84	4.73	2.95
	Company	1.82	2.86	9.27	4.60	16.10
6	Algorithm	3.03	3.68	2.26	4.37	2.61
	Company	2.97	3.58	11.60	4.25	15.39
7	Algorithm	3.77	4.29	2.63	4.75	3.15
	Company	3.68	4.17	11.96	4.62	14.60
8	Algorithm	2.62	3.59	1.96	4.73	2.50
	Company	2.56	3.48	10.16	4.64	15.03
9	Algorithm	2.81	3.73	2.30	4.57	2.92
	Company	2.71	3.58	10.52	4.44	15.21
10	Algorithm	3.72	4.24	2.55	4.65	2.93
	Company	3.66	4.15	11.66	4.59	14.49
11	Algorithm	3.50	4.18	3.09	4.76	3.11
	Company	3.45	4.09	11.29	4.64	14.50
12	Algorithm	3.26	3.99	2.45	4.73	3.25
	Company	3.22	3.92	10.53	4.65	14.66
13	Algorithm	4.19	4.53	3.15	4.78	3.14
	Company	4.09	4.38	13.54	4.67	15.83
14	Algorithm	3.88	4.43	2.09	4.80	2.58
	Company	3.85	4.35	12.32	4.75	14.20
15	Algorithm	4.00	4.50	2.37	4.81	3.10
	Company	3.97	4.44	11.77	4.71	13.70

Table 9
Percentage of services with maximum affinity (per week).

Week	Company	Algorithm	Week	Company	Algorithm
1	76.30%	86.24%	9	78.46%	90.09%
2	75.90%	85.00%	10	79.58%	87.70%
3	82.79%	90.27%	11	84.82%	90.91%
4	82.96%	86.21%	12	83.09%	88.69%
5	77.40%	89.41%	13	80.52%	91.05%
6	76.64%	86.00%	14	86.59%	91.35%
7	83.89%	91.27%	15	85.83%	91.48%
8	78.42%	86.44%			

respect to the overtime, the travel time, the duration of the dynamic break and the idle time (i.e., the total duration of the paid breaks).

Since the overtime, the travel time, the duration of the dynamic break and the idle time are associated with the cost and are in the same time units, all their global results have been aggregated in Fig. 18. So, Fig. 18 shows the mean weekly times per caregiver according to the results in all the weeks.

Both Table 10 and Fig. 18 evidence the good behaviour of the algorithm with respect to the results of the company. Then, in the figure it can be appreciated that the mean of the overtime, the idle time and the break time are drastically reduced if we employ the algorithm. However, the difference is negligible in the case of travel times. The table also shows similar results over the weeks.

A more detailed study of the overtime, the dynamic break, the idle time and the travel time can be found in Fig. 19. As sake of illustration, the four characteristics are shown in week 6.[3] Hence, Fig. 19(a)

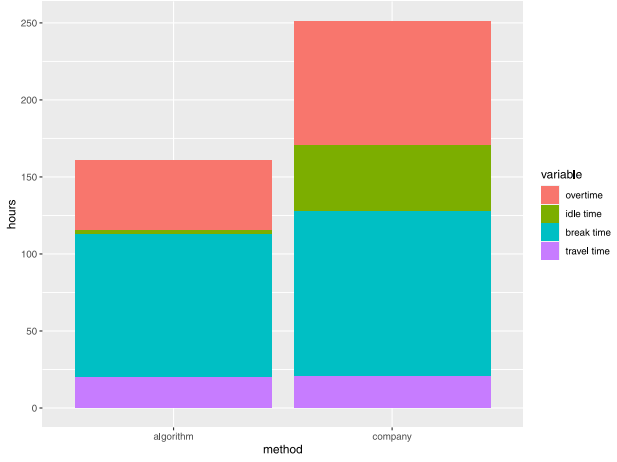


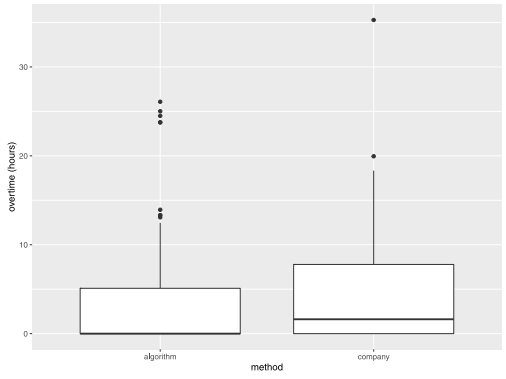
Fig. 18. Mean weekly times per caregiver (all weeks).

presents the comparison of the evolution of the overtime with respect to the agreed weekly working time for the different caregivers in both solutions, showing again the good performance of the algorithm. Figs. 19(b) to 19(d) make a comparison of both solutions on a daily basis. It is worth mentioning that there is a drastic reduction of the idle times every day (see Fig. 19(c)) . This reduction can be one of the key factors in reducing the overtimes. In terms of travel time, both solutions perform quite reasonably, with a median time of approximately 5 min every day and travel times that will never exceed 25 min.

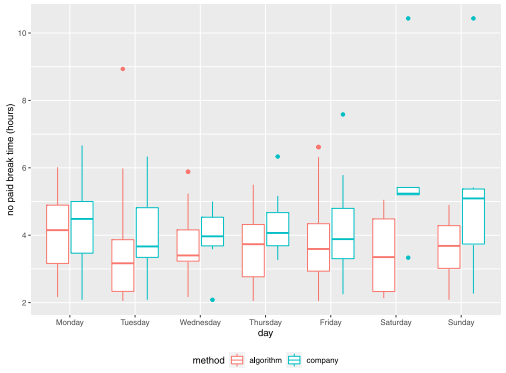
[3] A similar behaviour is observed in the other weeks.

Table 10
Computational results in hours for real instances (cost).

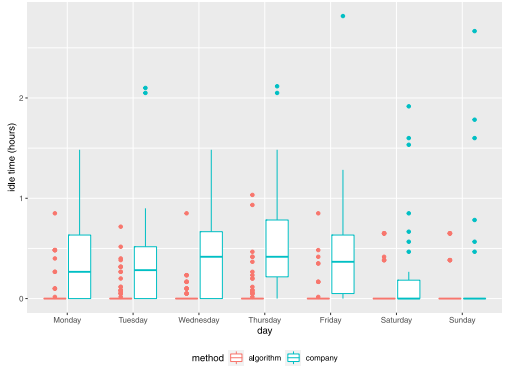
Week	Solution	Global			Per caregiver			
		Cost	Overtime	Worked time	Overtime	Travel time	Break	Idle time
1	Algorithm	1062.49	97.66	964.82	2.55	1.23	6.48	0.10
	Company	1290.31	211.38	1078.93	5.42	1.29	7.48	3.05
2	Algorithm	1110.62	131.21	979.40	3.48	1.34	6.67	0.13
	Company	1330.55	242.13	242.13	6.37	1.37	7.28	2.97
3	Algorithm	1187.72	192.33	995.38	5.58	1.51	8.17	0.32
	Company	1383.23	280.06	1103.16	8.24	1.54	7.70	3.41
4	Algorithm	1052.64	108.07	944.56	2.92	1.32	6.45	0.18
	Company	1247.58	203.93	1043.65	5.51	1.33	7.22	2.86
5	Algorithm	729.22	20.44	708.77	0.53	0.89	4.99	0.08
	Company	833.06	48.33	784.73	1.27	0.94	6.51	2.07
6	Algorithm	1028.19	129.28	898.90	3.59	1.30	4.79	0.12
	Company	1207.38	218.10	989.28	5.89	1.31	6.76	2.62
7	Algorithm	1132.08	156.60	975.47	4.48	1.53	6.52	0.21
	Company	1297.50	227.71	1069.78	6.51	1.53	8.22	2.90
8	Algorithm	881.07	43.12	837.94	1.05	1.08	5.30	0.16
	Company	1047.96	116.53	931.43	2.99	1.11	6.15	2.53
9	Algorithm	972.19	94.20	877.99	2.55	1.28	6.65	0.20
	Company	1162.40	192.08	970.31	5.49	1.33	8.53	2.75
10	Algorithm	1069.21	92.89	976.32	2.46	1.42	5.84	0.19
	Company	1286.75	206.65	1080.10	5.59	1.41	6.58	2.99
11	Algorithm	1045.20	111.42	933.78	3.20	1.43	5.98	0.08
	Company	1226.06	195.60	1030.46	5.29	1.38	7.12	2.71
12	Algorithm	980.80	75.60	905.19	1.94	1.31	5.68	0.13
	Company	1154.76	148.68	1006.08	3.81	1.34	6.21	2.72
13	Algorithm	1184.37	152.59	1031.78	4.06	1.53	6.14	0.16
	Company	1377.91	234.85	1143.06	6.18	1.47	7.13	3.18
14	Algorithm	1149.30	136.22	1013.08	3.39	1.42	6.34	0.22
	Company	1358.13	240.00	1118.13	6.32	1.44	6.87	3.01
15	Algorithm	1172.04	153.46	1018.58	4.18	1.52	6.58	0.23
	Company	1350.56	219.73	1130.83	5.94	1.50	7.47	3.29



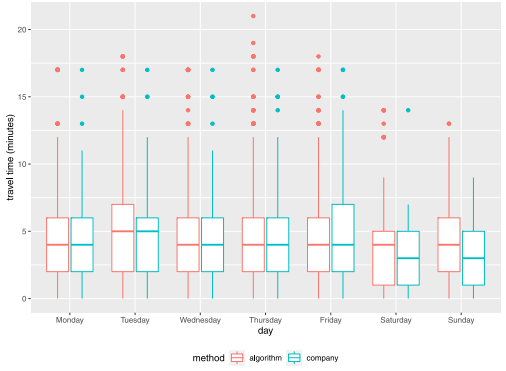
(a) Overtime for the caregivers.



(b) Dynamic break for the caregivers per day.



(c) Idle time for the caregivers per day.



(d) Travel time between services per day.

Fig. 19. Overtime, dynamic break, idle time and travel time (week 6).

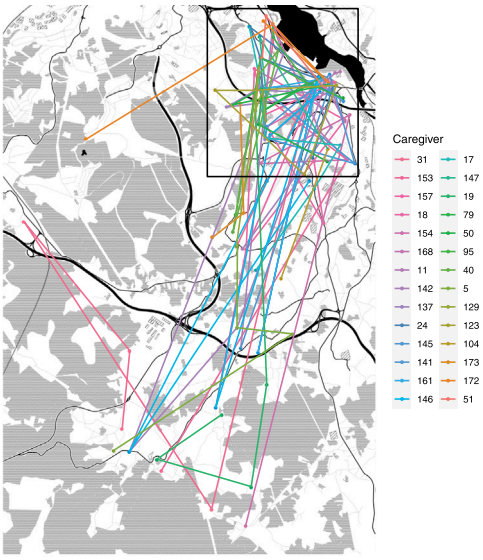


Fig. 20. Routes for a random day.

Finally, to illustrate the geographical distribution of the obtained routes for the caregivers, the routes provided by the solution on a random day are shown in Fig. 20. Analysing these routes, it can be seen that they cover two different areas: a rural and an urban one. The urban area (surrounded by the frame) is distinguished for having a lot of clustered services while the rural area has fewer services but spread over a more extensive area.

We selected 5 random caregivers who work on the urban area (Fig. 21(a)) and 5 caregivers who cover the rural area (Fig. 21(b)). It can be seen that the caregivers working in the rural area also have to carry out some urban services. This is necessary to balance the working time of the caregivers, because there are more services required in the urban area.

5. Concluding remarks

In this work, a real problem of a home care company is addressed. The problem shares many of the characteristics of other routing and scheduling problems in home care. But, although there is a huge literature devoted to the understanding of this type of problems, commonly known by the acronym HCSP, there is a special feature that substantially distinguishes it from the others and makes its study of great interest. In accordance with this characteristic, the longest break between two daily consecutive services of each caregiver will not be included as part of her working day, provided that it is greater than a fixed number of hours.

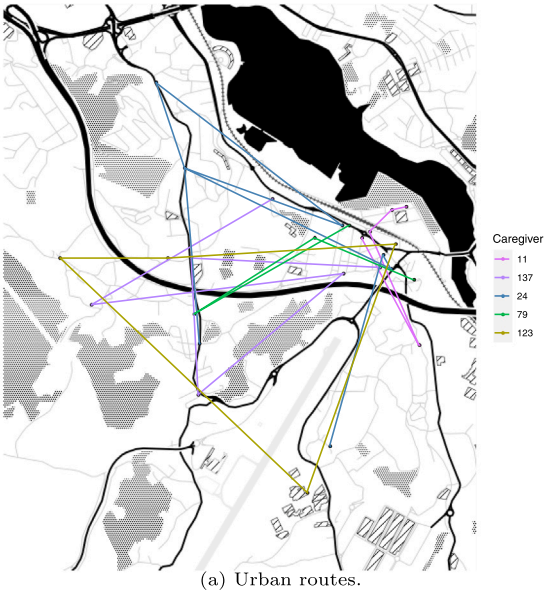
In order to gather all the requirements of the company, we formulated a MILP with two clearly differentiated objectives. The first and the most important objective is related to the welfare of the users, since improving the living conditions of the users is a crucial issue for the company. The second objective is to reduce the cost of the salaries of the caregivers as much as possible.

In a first attempt, we tackled the model with an optimization solver. However, due to the nature of the model, it only provided feasible solutions for 80% of the instances and guaranteed optimality in 45% of them.

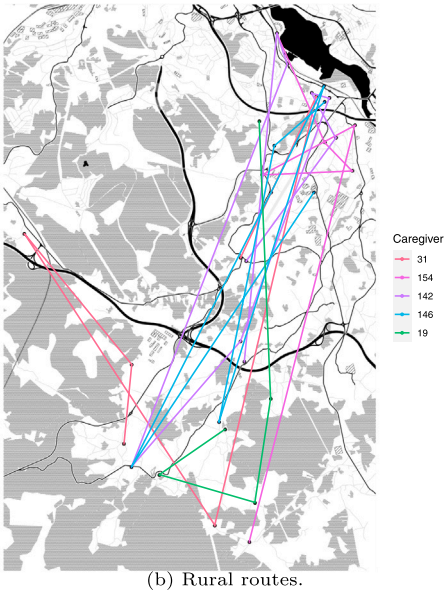
With the purpose of solving the problem in a more realistic setting, we developed an algorithm, based on the ALNS technique, that incorporates a new method for assessing the quality of the solutions. This method arises from the difficulty of determining the starting time of each service, once the services and the order of the services have been assigned to the caregivers. Basically, the method combines the margin of movement of the services within the hard and soft time windows with the grouping of services into blocks, thus allowing to manage the impact of the breaks between services. This is not an easy task, since many different scenarios might occur. Thus, it could be seen as the most innovative part of the framework.

Table A.11
List of tasks.

Task	Description	Task	Description
1	Get out of bed	9	Do the dishes
2	Make beds	10	Do the laundry
3	Cook breakfast	11	Cleaning
4	Bathe	12	Change diaper
5	Wash hair	13	Go for a walk
6	Dress/undress	14	Brushing teeth/dentures
7	Cook lunch	15	Cook dinner
8	Feed	16	Put to bed



(a) Urban routes.



(b) Rural routes.

Fig. 21. Routes separated into areas.

Table A.12
List of services.

Service	Tasks	User	Day	Hard TW	Soft TW	Duration
1	1, 2, 3, 4, 5, 6, 8	1	Mon	08:00–11:00	08:00–10:00	90
2	3, 8, 9, 10, 11, 12	9	Mon	09.30–12:30	09:30–11:30	120
3	13, 4, 5	10	Mon	11:30–14:00	12:00–14:00	60
4	7, 8, 9, 11	11	Mon	12:00–14:30	13:30–14:30	60
5	13, 15, 8, 9, 14, 16	1	Mon	18:00–21:00	18:00–20:00	120
6	3, 8, 11	2	Tue	08:00–09:00	08:00–09:00	60
7	1, 2, 3, 4, 5, 6, 8	1	Tue	08:00–11:00	08:00–10:00	90
8	4, 7, 10	12	Tue	10:30–14:00	10:30–11:30	60
9	13, 10	4	Tue	12:00–13:30	12:00–13:00	45
10	7, 8, 9, 11	11	Tue	12:00–14:30	13:30–14:30	60
11	13, 15, 8, 9, 14, 16	1	Tue	18:00–21:00	18:00–20:00	120
12	1, 2, 3, 4, 5, 6, 8	1	Wed	08:00–11:00	08:00–10:00	90
13	2, 11	3	Wed	09:30–10:30	09:30–10:30	30
14	3, 8, 9, 10, 11, 12	9	Wed	09.30–12:30	09:30–11:30	120
15	7, 8, 9, 11	11	Wed	12:00–14:30	13:30–14:30	60
16	13, 15, 8, 9, 14, 16	1	Wed	18:00–21:00	18:00–20:00	120
17	3, 8, 11	2	Tue	08:00–09:00	08:00–09:00	60
18	1, 2, 3, 4, 5, 6, 8	1	Thu	08:00–11:00	08:00–10:00	90
19	4, 7, 10	12	Thu	10:30–14:00	10:30–11:30	60
20	13, 10	4	Thu	12:00–13:30	12:00–13:00	45
21	7, 8, 9, 11	11	Thu	12:00–14:30	13:30–14:30	60
22	13, 15, 8, 9, 14, 16	1	Thu	18:00–21:00	18:00–20:00	120
23	1, 2, 3, 4, 5, 6, 8	1	Fri	08:00–11:00	08:00–10:00	90
24	3, 8, 9, 10, 11, 12	9	Fri	09.30–12:30	09:30–11:30	120
25	13, 4, 5	10	Fri	11:30–14:00	12:00–14:00	60
26	7, 8, 9, 11	11	Fri	12:00–14:30	13:30–14:30	60
27	13, 15, 8, 9, 14, 16	1	Fri	18:00–21:00	18:00–20:00	120
28	1, 2, 3, 8, 9	13	Mon	06:30–08:00	06:30–07:30	60
29	3, 8, 14, 9, 10, 12, 13	5	Mon	07:00–12:00	09:00–11:00	120
30	7, 8	14	Mon	12:00–14:00	12:00–13:00	30
31	13, 4, 5, 6, 12	6	Mon	16:00–20:00	16:00–20:00	120
32	10, 4, 5, 6	15	Mon	17:00–20:00	17:30–19:30	60
33	1, 2, 3, 8, 9	6	Tue	07:00–10:00	09:00–10:00	60
34	1, 2, 3, 8, 9	13	Tue	08:30–10:00	09:00–10:00	60
35	4, 5, 6, 12, 14, 11	7	Tue	10:00–13:00	10:00–12:00	90
36	5, 11	8	Tue	11:30–12:30	11:30–12:30	30
37	7, 8, 9, 14	16	Tue	11:30–14:00	11:30–13:30	60
38	7, 8, 14, 9, 12, 11	17	Tue	12:30–15:00	11:30–13:30	75
39	1, 2, 3, 8, 9	13	Wed	06:30–08:00	06:30–07:30	60
40	3, 8, 14, 9, 10, 12, 13	5	Wed	07:00–12:00	09:00–11:00	120
41	7, 8	14	Wed	12:00–14:00	12:00–13:00	30
42	13, 4, 5, 6, 12	6	Wed	16:00–20:00	16:00–20:00	120
43	10, 4, 5, 6	15	Wed	17:00–20:00	17:30–19:30	60
44	1, 2, 3, 8, 9	6	Thu	07:00–10:00	09:00–10:00	60
45	1, 2, 3, 8, 9	13	Thu	08:30–10:00	09:00–10:00	60
46	4, 5, 6, 12, 14, 11	7	Thu	10:00–13:00	10:00–12:00	90
47	5, 11	8	Thu	11:30–12:30	11:30–12:30	30
48	7, 8, 9, 14	16	Thu	11:30–14:00	11:30–13:30	60
49	7, 8, 14, 9, 12, 11	17	Tue	12:30–15:00	11:30–13:30	75
50	1, 2, 3, 8, 9	13	Fri	06:30–08:00	06:30–07:30	60
51	3, 8, 14, 9, 10, 12, 13	8	Fri	07:00–12:00	09:00–11:00	120
52	7, 8	14	Fri	12:00–14:00	12:00–13:00	30
53	13, 4, 5, 6, 12	6	Fri	16:00–20:00	16:00–20:00	120
54	10, 4, 5, 6	15	Fri	17:00–20:00	17:30–19:30	60

To evaluate the performance of the algorithm, we tested instances of 10, 15, 25, 50 and 100 services, by adapting to the characteristics of our problem the Solomon instances for routing and scheduling problems with time windows constraints. Finally, real data instances of 15 consecutive weeks from 2016 to 2017 were considered to compare the schedules provided by the company with the algorithm schedules. The results were very promising, since multiple configurations of the algorithm can be used to obtain good solutions for the Solomon instances, depending on their size. In terms of the real instances, the schedules obtained using the algorithm are much better than those provided by the company.

CRediT authorship contribution statement

Isabel Méndez-Fernández: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. **Silvia Lorenzo-Freire:** Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. **Ángel Manuel González-Rueda:** Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing.

Data availability

Data will be made available on request.

Acknowledgements

This research was funded by MICINN/AEI/10.13039/501100011033/ and ERDF/EU through R+D+I project grants MTM2017-87197-C3-1-P and PID2021-124030NB-C31 and by Consellería de Cultura, Educación e Universidades, Xunta de Galicia (Grupos de Referencia Competitiva ED431C-2020/14 and Centro de Investigación del Sistema universitario de Galicia ED431G-2019/01). Funding for open access charge: Universidade da Coruña/CISUG. Constructive comments from several referees and the editor are also gratefully acknowledged.

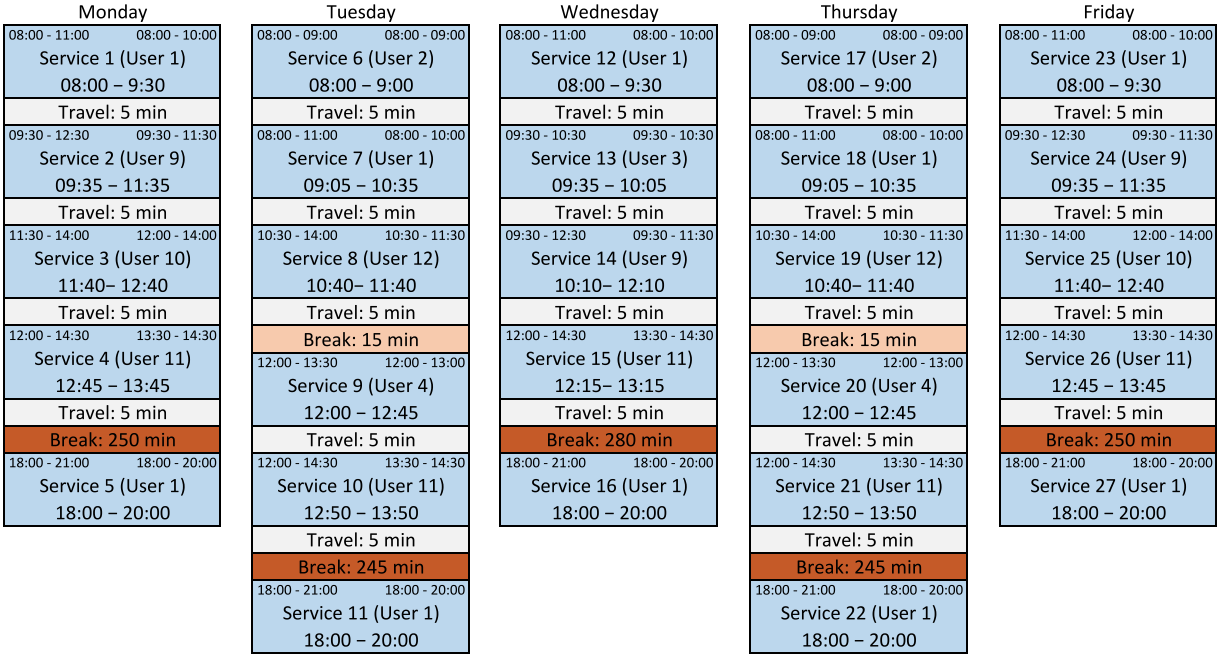


Fig. A.22. Schedules of Caregiver 1.

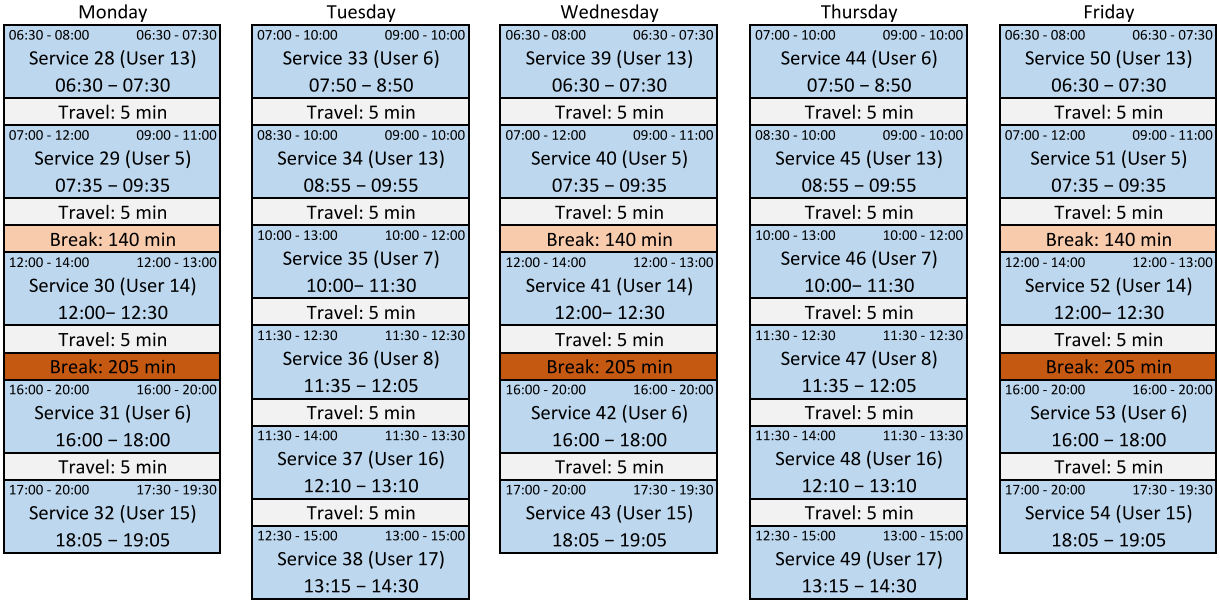


Fig. A.23. Schedules of Caregiver 2.

Appendix A. Example

In this section an example of the problem under study is presented. It consists of 54 services that two caregivers have to carry out over the course of a week.

First, Table A.11 shows a description of the tasks to be performed during the services. Then, Table A.12 describes the characteristics of the services including: the set of tasks assigned to each service, the user demanding the service, the day on which the service should be performed, the time windows and the duration. Finally, the affinity levels between users and the available caregivers are presented in Table A.13.

The weekly schedules of the caregivers are shown in Figs. A.22 and A.23. Each column represents the day schedule, the travel times and the breaks (the light ones are considered as working time, whereas the dark ones are unpaid breaks). Each box includes the following information: hard time window (top left), soft time window (top right), service and user (middle) and the scheduled start and end time (bottom).

Analysing the cost of the schedule for Caregiver 1, it can be seen that the schedule of Monday is repeated on Friday.[4] The same happens with Tuesday and Thursday. Since every day there is only one break of more than two hours, they will be subtracted from their corresponding working day. In this way, the fifteen minute break in Tuesday and Thursday belongs to the working day. In terms of Caregiver 2, the schedule of Monday is the same as the ones on Wednesday and Friday.

[4] Recurrent tasks are carried out at the same times.

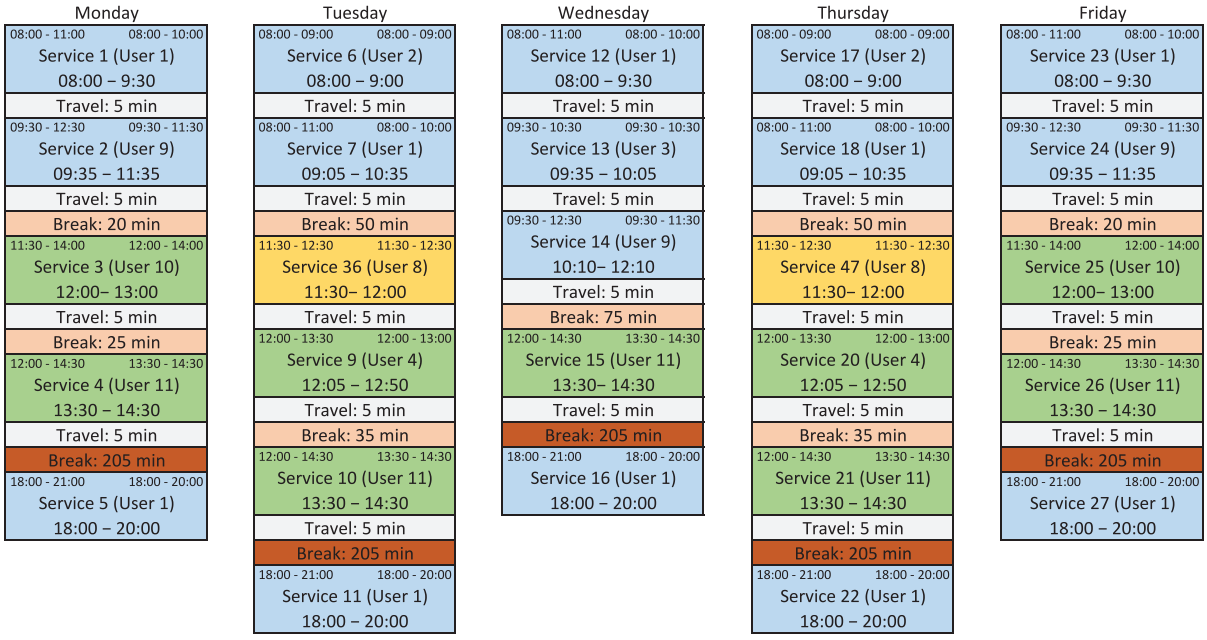


Fig. A.24. Improved schedules of Caregiver 1.

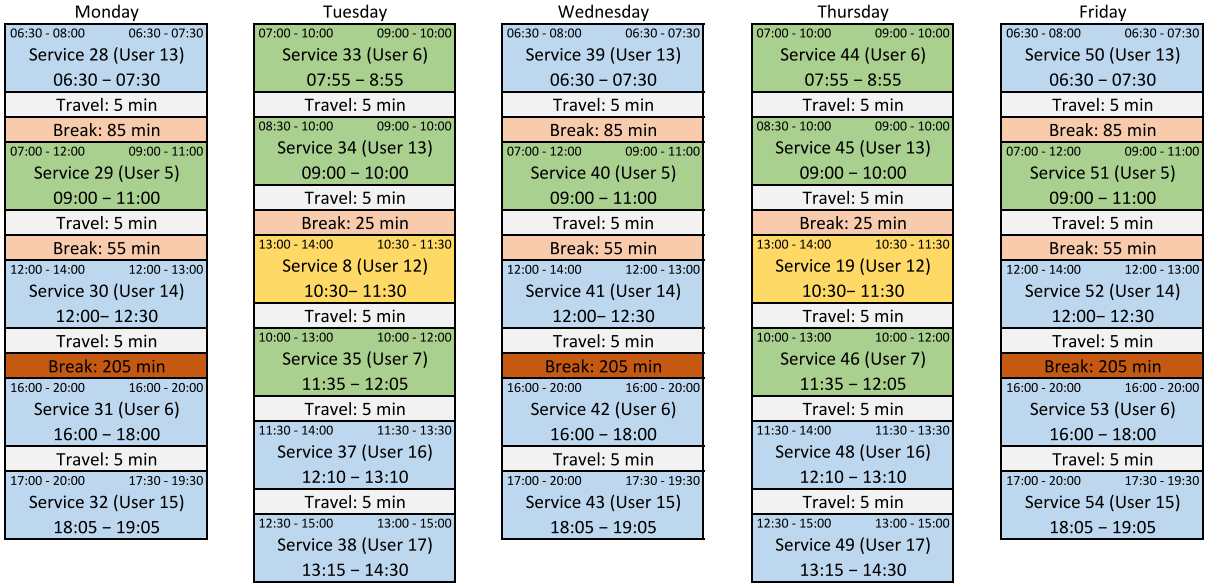


Fig. A.25. Improved schedules of Caregiver 2.

Table A.13
List of affinities.

User	Affinity Caregiver 1	Affinity Caregiver 2
1	5	2
2	2	2
3	5	2
4	4	1
5	4	4
6	2	5
7	2	4
8	4	2
9	4	2
10	5	4
11	4	2
12	2	4
13	4	5
14	1	5
15	2	5
16	2	5
17	2	2

Table A.14
Objective function values.

	Affinity	STW penalization	Overtime	Worked time
Caregiver 1	113	425	0	2330
Caregiver 2	118	405	50	2450
Total	231	830	50	4780

Table A.15
Improved objective function values.

	Affinity	STW penalization	Overtime	Worked time
Caregiver 1	117	120	175	2575
Caregiver 2	122	140	50	2440
Total	239	260	225	5015

During those days, Caregiver 2 has two breaks that last more than two hours, which means that only the largest one of them will not be paid. The schedule of Tuesday is repeated on Thursday and, in this case, Caregiver 2 has no breaks.

Table B.16
Mean RPD values comparing the algorithm with Gurobi.

	25%	50%	75%	100%	*auto_25%*	*auto_50%*	*auto_75%*	*auto_100%*
10_01	0.73	0	0	0	0.31	0	0	0
10_02	0.09	0	0	0	0	0	0	0
10_03	0.04	0	0	0.02	0.02	0.02	0.18	0
10_04	0	0	0	0	0	0	0	0
10_05	0.92	0	0	0.18	0.73	0.18	0	0
10_06	1.46	0.58	0	0	2.91	0	0	0
10_07	0	0	0	0	0.85	0	0	0
10_08	5.24	0	0	0	1.72	0	0	0
10_09	0.16	0	0	0	0.33	0	0	0
10_10	1.72	0	0	0	0	0	0	0
15_01	0	0	0	0	0.25	0	0	0
15_02	0	0	0	0	0.99	0	0	0
15_03	0.10	0	0	0	0.29	0	0	0
15_04	0	0	0	0	0	0	0	0
15_05	1.40	−1.26	−1.48	−1.04	0.22	−0.16	−0.93	−0.88
15_06	1.90	−1.49	−2.69	−2.33	−1.01	−1.07	−2.27	−2.67
15_07	0.62	0	−0.77	−0.62	0.10	0.26	−0.31	−0.31
15_08	1.26	−2.15	−2.31	−2.15	−0.38	−2.15	−2.31	−2.31
15_09	2.36	−3.23	−3.23	−3.23	−1.23	−3.23	−3.23	−3.23
15_10	2.48	0	0	0	5.86	0	0	0

Table B.17
RPD values comparing the algorithm with the best solution found.

	25%	50%	75%	100%	*auto_25%*	*auto_50%*	*auto_75%*	*auto_100%*
10_01	0	0	0	0	0	0	0	0
10_02	0	0	0	0	0	0	0	0
10_03	0	0	0	0	0	0	0	0
10_04	0	0	0	0	0	0	0	0
10_05	0	0	0	0	0	0	0	0
10_06	0	0	0	0	0	0	0	0
10_07	0	0	0	0	0	0	0	0
10_08	0	0	0	0	0	0	0	0
10_09	0	0	0	0	0	0	0	0
10_10	0	0	0	0	0	0	0	0
15_01	0	0	0	0	0	0	0	0
15_02	0	0	0	0	0	0	0	0
15_03	0	0	0	0	0	0	0	0
15_04	0	0	0	0	0	0	0	0
15_05	0	0	0	0	0	0	0	0
15_06	0	0	0	0	0	0	0	0
15_07	0	0	0	0	0	0	0	0
15_08	0	0	0	0	0	0	0	0
15_09	0	0	0	0	0	0	0	0
15_10	0	0	0	0	0	0	0	0

(*continued on next page*)

According to the affinity levels, all services are being carried out by their preferred caregivers except the ones of Users 12 and 8. These users are currently assigned to a caregiver with whom they have an affinity level of 2, but their affinity with the caregiver not attending them is 4. In terms of the soft time window penalization, some services are carried out within their soft time window (e.g. Services 6, 17, 35, 46 and 13), while others start before (e.g. Services 3, 4, 15, 25 and 26) or end after (e.g. Services 7, 8, 18 and 19) their soft time window.

The objective function values, in minutes, are shown in Table A.14. Both schedules are similar in terms of affinity, soft time penalization and working time, even though Caregiver 2 has 50 min of overtime (if the agreed weekly working time is 40 h).

The schedule can be modified in order to improve the affinity and soft time window penalization. Figs. A.24 and A.25 present the new schedules and Table A.15 their objective function values.

The overall affinity is improved by interchanging Services 8, 19, 36 and 47 between both caregivers (see yellow boxes). As far as the soft time window penalization is concerned, its value is greatly reduced by modifying the schedule of the services carried out outside their soft time window (see green boxes). However, it will possibly imply an increment of the duration of some breaks. For instance, the delay of Services 3 and 4 on Monday, creates two new breaks of 20 and 25 min for Caregiver 1, which are not deducted from the journey. This type of modifications in the schedule can increase the overtime, as it happens in this case for Caregiver 1, resulting now in 175 min of overtime.

Appendix B. Solomon instances tables

In this Appendix several tables show the performance of the algorithm with the Solomon instances.

Table B.16 presents, for the instances with 10 and 15 services, the mean RPD,[5] with respect to the second objective.[6] This value indicates the mean of the deviations of the solutions found by the algorithm (with 1000 iterations, considering the different values of p) from the solution obtained by using Gurobi solver.[7]

[5] For each instance, let x and x' be two solutions. The Relative Percentage Deviation (RPD), of x from x', is defined as $RPD(x) = \frac{f(x)-f(x')}{f(x')} \times 100$.

[6] There are no differences with respect to the first objective (welfare) for these instances.

[7] Note that, in this case, a negative RPD value indicates that in mean the algorithm finds a better solution than Gurobi, and a positive value indicates the contrary.

Table B.17 (*continued*).

	25%	50%	75%	100%	*auto*_25%	*auto*_50%	*auto*_75%	*auto*_100%
25_01	$2.83E{-3}$	0	0	0	$3.92E{-3}$	0	0	0
25_02	$8.34E{-3}$	$6.45E{-3}$	$6.45E{-3}$	$6.45E{-3}$	$3.02E{-3}$	$9.09E{-4}$	$7.27E{-4}$	$6.45E{-3}$
25_03	2.15	0	0	0	0	0	0	0
25_04	4.67	3.79	5.19	2.58	4.83	3.11	1.72	2.58
25_05	$9.89E{-1}$	$6.57E{-1}$	$6.74E{-1}$	$6.52E{-1}$	3.58	1.96	$3.35E{-1}$	$6.52E{-1}$
25_06	$1.22E{-3}$	$9.32E{-4}$	$7.29E{-4}$	$8.13E{-4}$	$6.44E{-4}$	0	0	$8.13E{-4}$
25_07	1.29	$3.24E{-1}$	$9.72E{-1}$	$6.43E{-1}$	5.82	2.57	1.29	$6.43E{-1}$
25_08	1.33	1.97	1.32	1.63	1.96	1.96	1.64	1.63
25_09	$2.03E{-5}$	0	0	0	$4.07E{-5}$	$2.03E{-5}$	$2.03E{-5}$	0
25_10	$3.69E{-4}$	0	0	0	0	$1.62E{-3}$	0	0
50_01	$2.89E{-3}$	$3.16E{-3}$	$4.66E{-3}$	$5.28E{-3}$	$6.49E{-4}$	$8.59E{-4}$	$6.17E{-4}$	$2.22E{-3}$
50_02	$6.52E{-1}$	$6.62E{-1}$	$5.26E{-1}$	1.47	$7.68E{-3}$	$1.68E{-1}$	$8.07E{-1}$	$3.45E{-1}$
50_03	0	0	0	0	0	0	0	0
50_04	$6.07E{-4}$	$1.01E{-3}$	$8.04E{-4}$	$1.44E{-3}$	$1.09E{-3}$	$1.17E{-3}$	$1.49E{-4}$	$2.51E{-4}$
50_05	$4.91E{-4}$	$4.95E{-4}$	$7.70E{-4}$	$1.16E{-3}$	$7.70E{-4}$	$3.97E{-4}$	$2.08E{-4}$	$4.54E{-4}$
50_06	$6.72E{-5}$	$2.69E{-5}$	$2.02E{-4}$	$3.97E{-4}$	$6.72E{-5}$	0	$2.02E{-5}$	$6.72E{-6}$
50_07	$1.61E{-3}$	$8.10E{-4}$	$6.29E{-4}$	$1.40E{-3}$	$8.02E{-4}$	$1.03E{-3}$	$3.62E{-4}$	$8.02E{-4}$
50_08	$3.22E{-4}$	$1.82E{-4}$	$4.41E{-4}$	$1.12E{-3}$	$1.18E{-3}$	$7.84E{-4}$	$3.08E{-4}$	$2.66E{-4}$
50_09	$4.03E{-4}$	$9.34E{-4}$	$1.84E{-3}$	$3.15E{-3}$	$1.67E{-3}$	$9.41E{-4}$	$7.40E{-4}$	$8.87E{-4}$
50_10	$9.63E{-4}$	$1.06E{-3}$	$2.30E{-3}$	$3.59E{-3}$	$1.53E{-3}$	$2.53E{-4}$	$6.82E{-4}$	$3.67E{-4}$
100_01	$8.91E{-1}$	1.29	1.93	2.98	$4.06E{-1}$	$6.43E{-1}$	1.21	1.93
100_02	$8.33E{-2}$	$9.70E{-1}$	1.65	2.21	$4.82E{-1}$	$5.65E{-1}$	1.13	1.81
100_03	0	0	0	0	0	0	0	0
100_04	2.57	8.46	7.30	8.08	$7.76E{-1}$	3.73	5.44	5.56
100_05	0	0	0	0	0	0	0	0
100_06	$1.16E{-4}$	$2.44E{-4}$	–	–	$8.93E{-5}$	$2.30E{-4}$	$2.72E{-4}$	$1.65E{-4}$
100_07	$3.22E{-4}$	$5.41E{-4}$	–	–	$3.19E{-4}$	$3.21E{-4}$	$4.23E{-4}$	$3.75E{-4}$
100_08	$2.30E{-5}$	$2.01E{-5}$	–	–	$1.04E{-5}$	0	$2.76E{-5}$	$3.45E{-5}$
100_09	$2.62E{-4}$	–	–	–	$1.13E{-4}$	$4.28E{-4}$	$4.09E{-4}$	$3.40E{-4}$
100_10	$6.78E{-6}$	0	$2.37E{-5}$	$5.08E{-5}$	0	0	0	0

According to the results, the behaviour of the algorithm in small instances is better for the configurations with a larger p value (75%, *auto*_75%, 100% and *auto*_100%). We can therefore deduce that in small instances it is advisable to focus on major changes in the solution that allow the solution space to be adequately explored. In fact, for $p \in \{75\%, auto\_100\%\}$, the RPD shows a better performance of the algorithm with respect to Gurobi solutions.

With respect to all the instances considered, it is important to highlight that, when Gurobi finds a solution the algorithm is able to find solutions of the same quality or even much better. Therefore, to explore the stability of the algorithm in each instance we have studied the relative deviation of the solutions of the algorithm with respect to its best solution, which coincides with the best solution found for the instance. Table B.17 shows the mean RPD value (first objective) for all the instances comparing the solutions obtained by the algorithm with the best solution found.

The best RPD values are obtained using different configurations of p, depending on the size of the instance considered. In view of the results reported in the table, it can be seen that the algorithm is very stable for the instances considered. In the instances with 10 and 15 services, all the tests performed result in the best solution. As the number of services increases, it can be seen that the configurations with smaller values of p are the most suitable. This trend is clearly reflected in the instances with 100 services, where $p \in \{25\%, auto\_25\%\}$ lead to acceptable solutions in all cases. However, for these instances, the configurations with $p \in \{75\%, 100\%\}$ do not even provide solutions in the set time.

References

Akjiratikarl, C., Yenradee, P., Drake, P.R., 2007. PSO-based algorithm for home care worker scheduling in the UK. Comput. Ind. Eng. 53 (4), 559–583.

Bachouch, R.B., Guinet, A., Hajri-Gabouj, S., 2011. A decision-making tool for home health care nurses' planning. Supply Chain Forum Int. J. 12 (1), 14–20.

Bard, J.F., Shao, Y., Jarrah, A.I., 2014. A sequential GRASP for the therapist routing and scheduling problem. J. Sched. 17 (2), 109–133.

Bertels, S., Fahle, T., 2006. A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. Comput. Oper. Res. 33 (10), 2866–2890.

Braekers, K., Hartl, R.F., Parragh, S.N., Tricoire, F., 2016. A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience. European J. Oper. Res. 248 (2), 428–443.

Cappanera, P., Scutellà, M.G., 2015. Joint assignment, scheduling, and routing models to home care optimization: A pattern-based approach. Transp. Sci. 49 (4), 830–852.

Carello, G., Lanzarone, E., 2014. A cardinality-constrained robust model for the assignment problem in Home Care services. European J. Oper. Res. 236 (2), 748–762.

Chaieb, M., Jemai, J., Mellouli, K., 2020. A decomposition - construction approach for solving the home health care scheduling problem. Health Care Manag. Sci. 23 (2), 264–286.

Cissé, M., Yalçındağ, S., Kergosien, Y., Şahin, E., Lenté, C., Matta, A., 2017. OR problems related to Home Health Care: A review of relevant routing and scheduling problems. Oper. Res. Health Care 13–14, 1–22.

Decerle, J., Grunder, O., Hajjam El Hassani, A., Barakat, O., 2019. A hybrid memetic-ant colony optimization algorithm for the home health care problem with time window, synchronization and working time balancing. Swarm Evol. Comput. 46, 171–183.

Erdem, M., Bulkan, S., 2017. A two-stage solution approach for the large-scale home healthcare routing and scheduling problem. South Afr. J. Ind. Eng. 28, 133–149.

Fikar, C., Hirsch, P., 2017. Home health care routing and scheduling: A review. Comput. Oper. Res. 77, 86–95.

Garaix, T., Gondran, M., Lacomme, P., Mura, E., Tchernev, N., 2018. Workforce scheduling linear programming formulation. IFAC-PapersOnLine 51, 264–269.

Grenouilleau, F., Legrain, A., Lahrichi, N., Rousseau, L.-M., 2019. A set partitioning heuristic for the home health care routing and scheduling problem. European J. Oper. Res. 275 (1), 295–303.

Gurobi Optimization, L., 2021. Gurobi Optimizer Reference Manual.

Kergosien, Y., Lenté, C., Billaut, J.-C., 2009. Home health care problem: An extended multiple traveling salesman problem. In: 4th Multidisciplinary International Conference on Scheduling: Theory and Applications. Dublin, Ireland, pp. 85–92.

Liu, R., Yuan, B., Jiang, Z., 2017. Mathematical model and exact algorithm for the home care worker scheduling and routing problem with lunch break requirements. Int. J. Prod. Res. 55 (2), 558–575.

Mankowska, D.S., Meisel, F., Bierwirth, C., 2014. The home health care routing and scheduling problem with interdependent services. Health Care Manag. Sci. 17 (1), 15–30.

Maya Duque, P.A., Castro, M., Sörensen, K., Goos, P., 2015. Home care service planning. The case of Landelijke Thuiszorg. European J. Oper. Res. 243 (1), 292–301.

Méndez-Fernández, I., Lorenzo-Freire, S., Garcí a Jurado, I., Costa, J., Carpente, L., 2020. A heuristic approach to the task planning problem in a home care business. Health Care Manag. Sci. 23 (4), 556–570.

Mosquera, F., Smet, P., Vanden Berghe, G., 2019. Flexible home care scheduling. Omega 83, 80–95.

Nickel, S., Schröder, M., Steeg, J., 2012. Mid-term and short-term planning support for home health care services. European J. Oper. Res. 219 (3), 574–587, Feature Clusters.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Comput. Oper. Res. 34 (8), 2403–2435.

Rest, K.-D., Hirsch, P., 2016. Daily scheduling of home health care services using time-dependent public transport. Flex. Serv. Manuf. J. 28 (3), 495–525.

Riazi, S., Wigström, O., Bengtsson, K., Lennartson, B., 2019. A column generation-based gossip algorithm for home healthcare routing and scheduling problems. IEEE Trans. Autom. Sci. Eng. 16 (1), 127–137.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472.

Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. 35 (2), 254–265.

Trautsamwieser, A., Hirsch, P., 2011. Optimization of daily scheduling for home health care services. J. Appl. Oper. Res. 3 (3), 124–136.

Van Rossum, G., Drake, F.L., 2009. Python 3 Reference Manual. CreateSpace, Scotts Valley, CA.