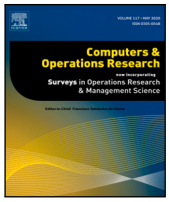
Computers & Operations Research 150 (2023) 106081

Contents lists available at ScienceDirect

Computers and Operations Research

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



A model-based evolutionary algorithm for home health care scheduling

Yoram Clapper [a,*], Joost Berkhout [a], René Bekker [a], Dennis Moeke [b]

[a] Department of Mathematics, Vrije Universiteit Amsterdam, The Netherlands
[b] Research Group Logistics and Alliances, HAN University of Applied Sciences, Arnhem, The Netherlands

ARTICLE INFO

Keywords:
OR in health services
Evolutionary algorithms
Routing
Scheduling

ABSTRACT

In this paper, for the first time a model-based evolutionary algorithm is presented for a real-life Home Health Care Routing and Scheduling Problem (HHCRSP). The algorithm generates routes consisting of care activities jointly with the underlying shift schedule, while taking into account the qualification levels. The performance is optimized in terms of travel time, time window waiting time and shift overtime. The algorithm is a novel extension of the permutation Gene-Pool Optimal Mixing Evolutionary Algorithm. Numerical experiments, using real-life data, show that the algorithm performs close to optimal for small instances, and outperforms schedules from a case study, leading to efficiency gains of 41%. Furthermore, it is shown that the model-based evolutionary algorithm performs better than a more traditional evolutionary algorithm, which demonstrates the importance of learning and exploiting a model to guide the optimization in HHCRSP.

1. Introduction

Dutch home care providers face the challenge of providing high-quality care while struggling with increasing demand and workforce shortages. This increase in demand is partly driven by the aging of the Dutch population. The percentage of people over 65 is expected to increase from 19% in 2020 to 25% in 2040, with the share of people aged 80 and over growing even more rapidly (NIDI/CBS, 2020). The prevalence of physical or mental disability, and thus the need for long-term care, increases with age. Furthermore, under pressure of health care reforms, nowadays care is provided as close to people's own living environment as possible. This has led to a decrease in the demand for nursing home care in favor of home care. In addition, more often home care services are advocated as an alternative to conventional (more expensive) hospital admissions. Overall, from the perspective of demand, it can be concluded that Dutch home care providers have to deliver services to a growing number of clients with a wider range of needs. From the supply perspective, the situation is as challenging as home care providers are confronted with an ever-growing labor shortage of healthcare professionals (ActiZ, 2021; SER, 2020).

As a consequence of these challenges, many Dutch home care providers are searching for ways to improve the balance between the available workforce capacity (i.e., supply) and the needs and preferences of their clients (i.e., demand). An appropriate balance is crucial for these clients due to their dependence on health services. According to the World Health Organization "home care" can be defined as an array of health and social support services provided to clients in their own residence (Knight and Tjassing, 1994). In the Netherlands, home care clients are in need of assistance with basic activities of daily living due to temporary or long-term physical or psychological disabilities. Consequently, in most cases, the required care or support has a direct impact on the daily routines of the clients. As such, the quality of care in a home care context largely depends on the coordination and timing of the service delivery.

In operational terms, the capacity in home care services relates to establishing shifts for care workers. Here, a *shift* specifies the start and end times as well as the required qualification level, and serves as a placeholder before an actual care worker is assigned. Typically, the assignment of care workers to shifts is done in a separate step, which is labeled as the nurse rostering problem. In practice, the collection of all shifts is often referred to as a *blueprint* and thus describes the type and amount of capacity required throughout time, without having the actual care workers assigned. The design of a blueprint (also referred to as *shift pattern* design) is one of the decision problems within the design of delivery of home health care, see Figure 1 of Grieco et al. (2021). The challenge in the construction of a blueprint is to create a shift pattern that matches the predicted workload as closely as possible, without exceeding the available staff capacity. An essential complication in home care services is that the workload is not a priori clear, as it depends on the routing of visits affecting travel times. The latter, i.e., routing of visits, has recently received increasing attention in the literature and has been labeled as the Home Health Care Routing and Scheduling Problem (HHCRSP). Hence, capacity decisions related to

* Corresponding author.
E-mail addresses: y.clapper@vu.nl (Y. Clapper), joost.berkhout@vu.nl (J. Berkhout), r.bekker@vu.nl (R. Bekker), dennis.moeke@han.nl (D. Moeke).

https://doi.org/10.1016/j.cor.2022.106081
Received 21 January 2022; Received in revised form 7 November 2022; Accepted 7 November 2022
Available online 13 November 2022

shift patterns need to be integrally taken with operational decisions concerning routing of visits, see Figure 1 of Grieco et al. (2021).

In practice, in a long-term care setting, designing shift patterns and routing of visits is often done manually and without a sound quantitative basis. This results in an inefficient planning process and suboptimal performance in terms of, e.g., service-level, workload balance and travel times (Bekker et al., 2019; Restrepo et al., 2020).

Since HHCRSP is NP-hard, one has to resort to metaheuristics to solve practically-sized instances in general. A particular class of metaheuristics is Evolutionary Algorithms (EAs) which are population-based, stochastic search algorithms inspired by mechanisms of natural evolution that are known to be effective in solving general NP-hard optimization problems (Eiben and Smith, 2015). In the last decades, so-called model-based EAs have been developed that are now seen as the most powerful type of EAs for discrete and continuous optimization (Cheng et al., 2018; Bosman et al., 2016). In contrast to traditional EAs and other typical "blind" metaheuristics, model-based EAs automatically learn a model that captures efficient subsolutions and which is exploited to guide the optimization. Under the plausible assumption that there is an exploitable structure, this will enhance the optimization (Bosman et al., 2016).

Inspired by the effectiveness of model-based EAs, we develop for the first time a model-based EA that can solve HHCRSP effectively. More specifically, our model-based EA is based on the recently developed *permutation Gene-pool Optimal Mixing Evolutionary Algorithm* (pGOMEA) from Bosman et al. (2016) that proved to be highly efficient for permutation problems such as flowshop scheduling. In this paper, pGOMEA is extended in order to solve HHCRSP, which in addition requires a task assignment. As such, we will refer to our extension as permutation-assignment GOMEA (paGOMEA) throughout this paper. The key aspect of the approach is that the stochastic search is driven by a linkage tree that is dynamically trained during the optimization (see Fig. 3 for an example). The linkage tree captures dependencies between variables that, in some sense, positively influence each other, for example, if three remote clients live near each other it is (most likely) effective to visit them sequentially (and maintain that when creating new solutions). To this end, a new dependency measure is developed based on information theory and statistical principles. The optimization becomes more effective when making the stochastic search biased towards these dependencies. In short, paGOMEA extends pGOMEA by (i) adding a shift assignment to the solution encoding and (ii) redefining the dependencies so that effective linkage trees are found for permutation assignment problems.

Our first contribution is that we make a first step in addressing the integrated problem of designing shift patterns and routing of visits during each shift, as it is specified by our partner HHC organization. In particular, specifying required capacity using a blueprint (i.e., design of shift patterns, specifying which type of care workers need to be scheduled at which moment) is not common in the HHC literature, whereas our objective function slightly differs from other HHCRSP studies. We illustrate that demand and capacity, as prescribed by the blueprint, are well balanced, and substantial improvements can be achieved compared to current practice.

Our second and main contribution is of a methodological nature by developing paGOMEA: the first model-based EA for HHCRSP that extends pGOMEA to solve permutation problems in a set partitioning context (it is therefore also applicable to, for example, Vehicle Routing Problems). Numerical experiments demonstrate that paGOMEA gives near optimal solutions for small instances and outperforms a more traditional EA that does not exploit a model during the optimization. Moreover, due to the black-box optimization nature of paGOMEA and its ability to learn a model on the fly, the algorithm is flexible which allows to quite easily adapt it to new real-life situations or workforce innovations (e.g., providing care at a distance via video calls).

The remainder of this paper is structured as follows. Section 2 presents an overview of related literature on HHCRSP and pGOMEA. In Section 3 we formulate the mathematical model of the HHCRSP together with the objective in regard to the optimization, and follow up in Section 4 with the MILP formulation; the MILP acts as a benchmark to validate our paGOMEA on small scale problems. Thereafter, in Section 5 the core of the paper is presented where we carefully describe paGOMEA. Finally, in Section 6 numerical experiments using real-life data from a Dutch home care provider are examined to analyze the performance of paGOMEA and to gain insight into the potential added value for practice. The paper is concluded in Section 7 with some discussions and directions for future work.

2. Literature

In this section, we first consider the literature related to HHCRSP and then consider the literature related to pGOMEA.

The first publication on routing and scheduling of home care workers can be traced back to the late nineties (Begur et al., 1997). In this early study, a case study is used to describe an integrated spatial decision support system (SDSS) that provides support with the assignment of home care workers to visits and determining the sequence in which the visits should be carried out. Since then, a large number of studies have been published with a wide variety of objectives and constraints. This diversity in HHCRSP formulations makes it difficult to compare existing methods. Therefore, we have chosen to take three relatively recent literature studies as a starting point for the overview presented in this section: Cissé et al. (2017), Di Mascolo et al. (2021) and Fikar and Hirsch (2017). Next to routing and scheduling, another interesting paper is (Grieco et al., 2021), which provides a more generic literature review of operations research applied to decisions in home health care.

Levels of planning. The planning and scheduling of home care workers can be categorized as strategic, tactical and/or operational (Cissé et al., 2017). On a strategic level, the focus mainly lies on where to locate home health care facilities across a geographical territory (i.e., the home health care location problem) (e.g., Rodriguez-Verjan et al. (2018)) and partitioning (i.e., districting) the geographical coverage area of a specific home care provider (e.g., Benzarti et al. (2013)). For decision making on a tactical level, the main objectives are (1) matching the size and composition of the home care teams with the local demand (i.e., dimensioning), see e.g. Restrepo et al. (2020) and Rodriguez et al. (2015), and (2) determining shift patterns (start and end times, breaks, etc.), along with assigning the number and type of care workers to each shift (i.e., shift scheduling). Finally, on an operational level the focus is on tasks to be performed on weekly or daily bases. More specifically, this concerns assigning care tasks to the available working shifts or specific care workers (i.e., the operator assignment problem) and the determination of the timing and order in which the care tasks should be carried out (HHCRSP), see, e.g., Bredström and Rönnqvist (2008) and Rasmussen et al. (2012). Most of the available studies focus on HHCRSP, whereas studies paying attention to the strategic and/or tactical challenges are less widely available. In particular, integrated staffing and scheduling problems for home healthcare have rarely been studied in the literature (Restrepo et al., 2020). More recently, some studies have considered the assignment of staff combined with routing for related problems. For instance, Gu et al. (2022) consider the Workforce Scheduling and Routing Problem (WSRP) in which a set of service providers need to complete tasks at different locations. The combination of tactical and operational decisions is also addressed in Schrotenboer et al. (2018) where technicians can be shared between wind farms, next to the construction of the daily routes. Alp and Alkaya (2022) consider vehicle routing problems (VRPs) in the presence of shift scheduling, which they label as `VRP_SA`. The authors mention that they bring together VRPs and workforce scheduling in a way that has not been done before. We note that their algorithm is based on EAs.

The different levels of planning reveal the need to decompose the complex planning process within health organizations into manageable

proportions (Dieleman et al., 2022; Hans et al., 2012; Matta et al., 2014). In this framework, the shift patterns play a fundamental role as they should balance the nursing capacity with the health care demand. However, it is not always evident how shifts are addressed in the HHCRSP literature. The two studies most closely related to ours are (Decerle et al., 2019; Rest and Hirsch, 2016) taking shift patterns during a single day into account as well. Specifically, Rest and Hirsch (2016) also use shifts to cover the peaks in demand during the morning, noon, and evening; similar peaks are also mentioned in Akjiratikarl et al. (2007). However, the focus in Rest and Hirsch (2016) is mainly on employees rather than on client preferences, whereas time-dependent travel times due to public transport are a crucial element. In Decerle et al. (2019) shifts are primarily used to balance the workload among caregivers. In Nickel et al. (2012), a two-stage approach is used to combine the operational planning problem with the nurse rostering problem. The emphasis is on nurses, and not so much on the shift structure, and a nurse-shift is determined by the weekday rather than the shift pattern across the day. The studies (Grenouilleau et al., 2019, 2020) use a scheduling horizon of a week, taking working time contracts into account. These papers use shifts across the day in terms of providers' time windows, but these seem endogenously given and do thus not construct a shift pattern. Similarly, integrated routing and scheduling of the health care staff is also considered in Yadav and Tanksale (2022), with the special feature of minimizing person-to-person contact in view of the risk of exposure to possibly contagious interactions.

In this study, we consider operational planning, whereas we additionally balance capacity with demand by determining shift patterns. As such, we also make a first step towards decision making at a tactical level. Our focus is on required capacity in terms of shift patterns, which is independent of the assignment of specific care workers. This differs from the limited number of papers on integrated routing and shift scheduling problems, where actual care workers are scheduled.

Objectives. The literature studies of Cissé et al. (2017), Di Mascolo et al. (2021) and Fikar and Hirsch (2017) provide comprehensive overviews of commonly used objectives and performance measures. Our objective is to minimize (i) total travel time, (ii), staff overtime, and (iii) time window violations of the client, which are also included in Table 6 of Di Mascolo et al. (2021) that provides an extensive overview of objective functions. We like to emphasize that (iii) is crucial for client-centered care, which is often lacking both in practice and in the literature. In fact, Mısır et al. (2015) is the only paper including all of the above three elements in the objective function. However, Mısır et al. (2015) consider a more abstract setting and apply hyper-heuristics based on a set of low-level heuristics. Furthermore, papers containing two out of (i)–(iii) in the objective function are (Bertels and Fahle, 2006; Decerle et al., 2018, 2019; Grenouilleau et al., 2019; Lin et al., 2018; Liu et al., 2018; Mankowska et al., 2014; Nasir and Dang, 2018; Nickel et al., 2012; Rest and Hirsch, 2016; Zhan and Wan, 2018). We refer to Di Mascolo et al. (2021) for an excellent overview of objectives considered in the literature.

To summarize, the combination of the three elements of our objective function differs from most literature, and allows us to combine efficiency with both staffing and client-friendly schedules.

HHCRSP extensions. We note that the majority of the HHCRSP focus on single-period optimization problems, i.e., settings where a single working day is assumed as the planning horizon (Fikar and Hirsch, 2017). However, extensions to longer time horizons can be found in e.g. Grenouilleau et al. (2019), Restrepo et al. (2020) and Trautsamwieser and Hirsch (2014). Moreover, extensions of HHCRSP to situations involving randomness can be found in Lanzarone and Matta (2014) and Restrepo et al. (2020) and Yuan et al. (2015), where the demand and service times are random, respectively. In Oladzad-Abbasabady and Tavakkoli-Moghaddam (2022) the compatibility between client and care worker is also considered, in addition to unexpected events and a longer time horizon. Furthermore, real-life situations are dynamic, due to cancellation of appointments or the last-minute absence of care workers. Examples of studies that consider a dynamic environment are (Bennett and Erera, 2011; Hewitt et al., 2016; Nguyen et al., 2015). Although we do not consider such features in our setting, we do want to mention that the black-box nature of our model-based EA gives the flexibility to optimize all kinds of objective functions (not only the one optimized in this paper).

EA and pGOMEA. The HHCRSP is an NP-hard problem that is often solved with (meta)heuristics. Our focus is on EAs, see for example Eiben and Smith (2015), which have shown to be effective to solve NP-hard problems having features related to permutations (de Oliveira da Costa et al., 2018b,a; Karakatič and Podgorelec, 2015). The last two decades have shown that so-called model-based EA, which learn and exploit a model to make the optimization more intelligent, are the best type of EAs for discrete and continuous optimization problems (Przewozniczek et al., 2021; Cheng et al., 2018). Only more recently in Bosman et al. (2016) and Aalvanger et al. (2018), it was also found that it is effective for permutation problems such as in routing and scheduling. Our method builds upon (Bosman et al., 2016) for permutation problems, which considers the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) from Bosman and Thierens (2013) that along the way learns a Linkage Tree (in short LT, also indicated as linkage model in the literature) to guide the optimization. We use pGOMEA to refer to this algorithm from Bosman et al. (2016), but it is also known in the literature as LT-GOMEA (Przewozniczek et al., 2021) and Linkage Tree Genetic Algorithm (LTGA) (Thierens, 2010). An attractive feature of pGOMEA is that the linkage tree is rebuilt during optimization which provides flexibility to adapt to new situations (Thierens and Bosman, 2012).

To conclude, this study extends pGOMEA from Bosman et al. (2016) by incorporating assignments in permutation problems and developing new models tailored to permutation assignment problems.

3. Problem description

In this section we introduce the model for an HHCRSP. The starting point is a given set of (indexed) activities of size $N \in \mathbb{N}$, denoted with $\mathcal{N} := \{1, 2, \ldots, N\}$. Each activity corresponds to a client, located in a given region, which requires service during one of the shifts. Here a shift is a placeholder for an employee with a specific qualification level. At a later stage, specific employees (with the proper qualification) need to be assigned to the shifts to make the shift schedule operational. The set of shifts is a set of size $V \in \mathbb{N}$, and is denoted with $\mathcal{V} := \{1, 2, \ldots, V\}$.

Each activity requires a minimum level of qualification. For this reason we introduce the qualification matrix Q of size $N \times V$, such that $Q_{n,v} = 1$ if shift $v \in \mathcal{V}$ is qualified to perform activity $n \in \mathcal{N}$, and $Q_{n,v} = 0$ otherwise. It is not allowed to schedule activity n during shift v if $Q_{n,v} = 0$. Finally, in the region there is a base location from which employees operate, i.e., the shifts start and end at the base location. For convenience, we will refer to the base location as activity 0; accordingly we define $\mathcal{N}_0 := \{0\} \cup \mathcal{N}$. To go from activity n to activity m there is a travel time of $d_{nm} \geq 0$ units of time, for $n, m \in \mathcal{N}_0$, where in case $n = m$ we set $d_{nm} = 0$.

Recall that a shift is a placeholder for an employee with a specific qualification level and indicates a start and end time during which the employee has to operate; these start and end times are either given or optimized. For shift $v \in \mathcal{V}$ the start time of the shift is denoted by $t_{v0} \geq 0$ and the (intended) shift duration is denoted by $u_v \geq 0$ (and implies the end time). A shift starts at the determined start time, whereas shift overtime is allowed, which is defined as the time difference between the return time to the base location and the end time, provided it is non-negative.

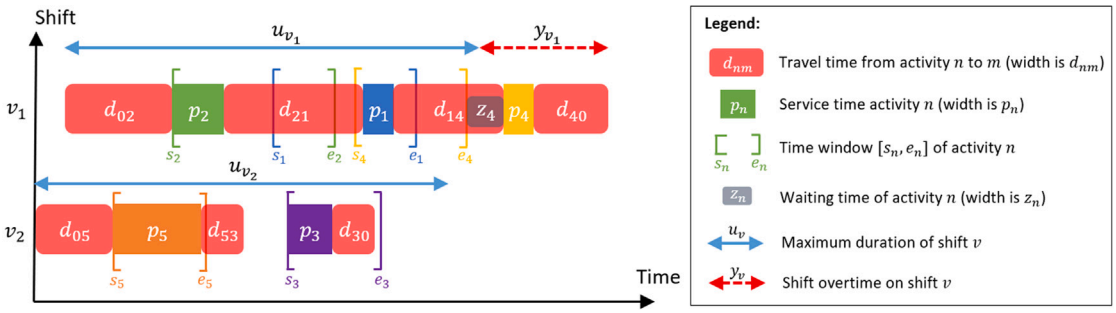


Fig. 1. Example with activities $\mathcal{N} = \{1, 2, 3, 4, 5\}$ and shifts $\mathcal{V} = \{v_1, v_2\}$ for which the following schedule is plotted as a Gantt chart: shift v_1 goes to activities $2 \rightarrow 1 \rightarrow 4$ starting at time $s_2 - d_{02}$, and shift v_2 goes to activities $5 \rightarrow 3$ starting at time 0.

Every activity has a time window defined by a start time $s_n \geq d_{0n}$ and a due time $e_n \geq s_n$, for activity $n \in \mathcal{N}$. The start and due times mark the points in time between which activity n preferably starts. Similar to the shifts, it is not allowed to start the activity before the start time, but tardiness is allowed and will be referred to as waiting time. The time it takes to complete an activity is called the service time and is denoted by $p_n \geq 0$ for activity $n \in \mathcal{N}_0$, here we take $p_0 = 0$.

The goal is to assign the activities to the available shifts so that each activity is covered exactly once by exactly one shift of the required qualification level. It is not necessary to use all shifts, that is, it is allowed to have no activities assigned to a shift. The collection of shifts that are carried out provide the shift pattern or blueprint, which play an important role at the tactical level. Thus, a schedule consists of the following three elements: (i) a shift schedule containing start and end times and qualification levels of each shift, (ii) activities that are assigned to the shifts, and (iii) planned arrival times for each activity. We seek to find a schedule that minimizes the cumulative travel time over all selected shifts, as well as the total (time window) waiting time and shift overtime. To illustrate most problem aspects, Fig. 1 plots the Gantt chart of an example schedule. An overview of the notation can be found in Table 1 in the next section.

4. MILP formulation

In this section we formulate the problem as described in Section 3 as a Mixed-Integer Linear Programming (MILP) problem.

4.1. Decision variables

To formulate the MILP let us first introduce the decision variables. Let the binary variables x_{vnm}, $v \in \mathcal{V}$, $n, m \in \mathcal{N}_0$, with $n \neq m$ unless $n = m = 0$, yield the value 1 if activity m directly follows activity n on shift v, and 0 otherwise; here $x_{v00} = 1$ represents that shift v is not used. Denote $t_{vn} \geq 0$, $v \in \mathcal{V}$, $n \in \mathcal{N}_0$ for the arrival time at activity n on shift v. As mentioned in Section 3, the decision variable t_{v0} represents the start time of shift v. Furthermore, we introduce $t_{v,N+1} \geq 0$, $v \in \mathcal{V}$, as the return time to the base location for shift v. Note that the actual shift duration $t_{v,N+1} - u_v$ of shift v may differ from the intended shift duration u_v and possibly results in shift overtime when this happens. Let $y_v \geq 0$, $v \in \mathcal{V}$, be the shift overtime of shift v, and let $z_n \geq 0$, $n \in \mathcal{N}$, be the waiting time of activity n. We refer to Table 1 for an overview of the sets, parameters and decision variables of the model.

4.2. Formulation

The MILP is as follows: minimize the weighted objective function, i.e., the weighted sum over the cumulative travel time, total shift overtime and total waiting time

$$w_x \sum_{v \in \mathcal{V}} \sum_{n \in \mathcal{N}_0} \sum_{m \in \mathcal{N}_0 \setminus \{n\}} d_{nm} x_{vnm} + w_y \sum_{v \in \mathcal{V}} y_v + w_z \sum_{n \in \mathcal{N}} z_n, \tag{1}$$

where $w_x, w_y, w_z \geq 0$, subject to the constraints described below:

$$\sum_{m \in \mathcal{N}_0} x_{v0m} = \sum_{n \in \mathcal{N}_0} x_{vn0} = 1, \qquad \forall v \in \mathcal{V}, \tag{2}$$

$$\sum_{m \in \mathcal{N}_0 \setminus \{n\}} x_{vmn} = \sum_{m \in \mathcal{N}_0 \setminus \{n\}} x_{vnm}, \qquad \forall v \in \mathcal{V}, n \in \mathcal{N}, \tag{3}$$

$$\sum_{v \in \mathcal{V}} \sum_{m \in \mathcal{N}_0 \setminus \{n\}} x_{vmn} = \sum_{v \in \mathcal{V}} \sum_{m \in \mathcal{N}_0 \setminus \{n\}} x_{vnm} = 1, \quad \forall n \in \mathcal{N}. \tag{4}$$

$$\sum_{m \in \mathcal{N}_0} x_{vnm} \leq Q_{n,v}, \qquad \forall v \in \mathcal{V}, n \in \mathcal{N}. \tag{5}$$

The constraints in (2) ensure that every shift starts and ends at the base location. The constraints in (3) ensure that the (non-)arrival at an activity on a shift implies a (non-)departure from the activity on the shift (and the other way around). The constraints in (4) ensure that every activity is attended to exactly once on exactly one shift. The constraints in (5) ensure that the qualification requirements are met. Moreover, we need the following constraints specifying the arrival time for each activity

$$t_{vn} + p_n + d_{nm} \leq t_{vm} + M(1 - x_{vnm}), \qquad \forall v \in \mathcal{V}, n \in \mathcal{N}_0, m \in \mathcal{N} \setminus \{n\}, \tag{6}$$

$$t_{vn} + p_n + d_{n0} \leq t_{v,N+1} + M(1 - x_{vn0}), \quad \forall v \in \mathcal{V}, n \in \mathcal{N}. \tag{7}$$

The constraints in (6) ensure that the arrival times are attainable and in consecutive order (and additionally eliminates sub-tours), whereas the constraints in (7) define the decision variable $t_{v,N+1}$. Here $M \geq 0$ is chosen sufficiently large, e.g., $M = \sum_{v \in \mathcal{V}} u_v + \sum_{n \in \mathcal{N}} p_n + \sum_{n,m \in \mathcal{N}_0} d_{nm}$. Next, the following constraints relate the arrival times at the activities to the time windows and shifts.

$$t_{v0} = \sum_{n \in \mathcal{N}} (s_n - d_{0n}) x_{v0n} \quad \forall v \in \mathcal{V}, \tag{8}$$

$$t_{vn} \geq s_n, \qquad \forall v \in \mathcal{V}, n \in \mathcal{N}, \tag{9}$$

$$t_{vn} \leq e_n + z_n, \qquad \forall v \in \mathcal{V}, n \in \mathcal{N}, \tag{10}$$

$$t_{v,N+1} \leq t_{v0} + u_v + y_v, \qquad \forall v \in \mathcal{V}. \tag{11}$$

The constraints in (8) fix for each shift the shift start time to the time window start time of the first activity on the shift (taking travel time into account). Here we note that the constraints in (8) are added to make a fair comparison with paGOMEA regarding the numerical results (see Section 6); by removing the constraints in (8) the start times are chosen optimally. Do note that travel times are uncertain in practice, and adding constraints (8) can therefore lead to more robust solutions. The constraints in (9) ensure that the service can only start after the start time of the time window. The constraints in (10) and (11) define the waiting times and shift overtimes, respectively. Finally, we have the

Table 1
Problem and MILP notation overview.

Sets	
$\mathcal{N}$	Set of activities indexed from 1 to N.
$\mathcal{N}_0$	Set of activities and base location; $\mathcal{N}_0 = \{0\} \cup \mathcal{N}$.
$\mathcal{V}$	Set of shifts indexed from 1 to V.
Parameters	
Q	Qualification matrix; $Q_{n,v} = 1$ if shift v is qualified for activity n, and $Q_{n,v} = 0$ otherwise, for $n \in \mathcal{N}, v \in \mathcal{V}$.
d_{nm}	Travel time from activity n to m, $n, m \in \mathcal{N}_0$.
$[s_n, e_n]$	Time window of activity $n \in \mathcal{N}$.
p_n	Service time of activity $n \in \mathcal{N}_0$.
u_v	Intended duration of shift $v \in \mathcal{V}$
Decision variables	
x_{vnm}	Binary variable equal to 1 if activity m directly follows n on shift v, and 0 otherwise, for $n, m \in \mathcal{N}_0$ with $n \neq m$ unless $n = m = 0$.
t_{vn}	Arrival time at activity $n \in \mathcal{N}_0 \cup \{N+1\}$ on shift $v \in \mathcal{V}$, here t_{v0} and $t_{v,N+1}$ are the start and (actual) end time of shift v.
y_v	Shift overtime on shift $v \in \mathcal{V}$.
z_n	Waiting time of activity $n \in \mathcal{N}$.

following constraints to define the domain of the decision variables.

$$\begin{aligned} x_{vnm} &\in \{0,1\}, && \forall v \in \mathcal{V}, n, m \in \mathcal{N}_0 \text{ with } n \neq m \text{ unless } n = m = 0, \\ t_{vn} &\geq 0, && \forall v \in \mathcal{V}, n \in \mathcal{N}_0 \cup \{N+1\}, \\ y_v &\geq 0, && \forall v \in \mathcal{V}, \\ z_n &\geq 0, && \forall n \in \mathcal{N}. \end{aligned} \tag{12}$$

Remark 1. The MILP formulation can be easily modified to incorporate various other practical aspects. For instance, in this formulation only the start time of each shift is determined. However, it is possible to model the shift duration u_v as a decision variable without losing linearity of the model. For example, one can do this by imposing a lower and upper bound, or in other words, a minimal and maximal duration, on the shift duration (i.e., $lb \leq u_v \leq ub$, for $v \in \mathcal{V}$ and $0 \leq lb \leq ub$) and add the constraint: $\sum_{v \in \mathcal{V}} u_v \leq C$, where the constant $C > 0$ imposes a budget on the amount of time units used over all shifts. Such a budget C typically relates to decisions at the tactical or strategic level.

5. Permutation-assignment GOMEA

This section will introduce paGOMEA for permutation assignment problems by extending pGOMEA from Bosman et al. (2016). Both are distinguished from a standard EA by using a dynamically learned linkage tree to guide the evolution. Just as in a standard EA, a population is maintained of solutions/schedules, represented by encodings, that improves over generations. New candidate solutions are generated by recombining the encodings of two solutions, and better solutions form new generations. To increase the chance of finding better solutions, the recombination is specified by a linkage tree that is dynamically learned during the optimization. This linkage tree-based recombination ensures that efficient solution structures have a greater chance of remaining intact when varying solutions. For example, when activities A, B and C are often assigned in order $A \rightarrow B \rightarrow C$ on a shift because of efficiency, then it becomes more likely that this subsolution will be varied as a whole instead of breaking down this subsolution. After a user-defined termination condition is reached, the best solution in the current population is returned.

To extend pGOMEA from Bosman et al. (2016) to paGOMEA, the encoding needs to be changed to capture the shift allocation decision, and the construction of the linkage tree needs to be revised for effective recombinations of permutation assignment solutions. And although the focus of this study is on a home-health care setting, we would like to emphasize that the ideas in this section are also relevant to permutation assignment problems in general.

In the upcoming subsections, all the above elements from paGOMEA will be detailed, including the encoding, fitness function, population, recombination of solutions (including the linkage tree), the evolution of the population, and termination conditions, respectively. In Section 5.7 all elements will be combined to present the pseudo-code of paGOMEA given in Algorithm 1. Throughout this section, we use the running example from Fig. 2 to illustrate paGOMEA.

5.1. Encoding of solutions

The encoding provides a solution representation that captures all feasible schedules and should be convenient for the evolutionary search. An encoding for a schedule is a vector $r = (r_1, r_2, \ldots, r_N) \in (1, V+1)^N$ consisting of N keys corresponding to all activities. For each activity $n \in \mathcal{N}$, the key $r_n = v_n + u_n \in (1, V+1)$ indicates that activity n is scheduled on shift $v_n \in \mathcal{V} = \{1, \ldots, V\}$ with a relative "urgency" of $u_n \in (0,1)$, where a relatively small u_n indicates urgency and a relatively large u_n indicates no-urgency. The shift schedule can be found by sorting the urgency keys from small to large for activities on a particular shift. For example, the keys $(1.33, 1.05, 2.36, 1.12, 2.42, 3.59, 3.88)$ can be decoded to the schedule: activities $2 \rightarrow 4 \rightarrow 1$ on shift 1, activities $3 \rightarrow 5$ on shift 2, and activities $6 \rightarrow 7$ on shift 3. More examples can be found in Fig. 2.

Using this encoding, one can easily generate random feasible schedules. To that end, let $\mathcal{V}_n$ denote the qualified shifts for activity n. Then a random key r_n can be generated by drawing a random number v_n from $\mathcal{V}_n$ and drawing a random number u_n from $(0,1)$ and setting $r_n = v_n + u_n$. This way, the random keys of all activities can be drawn to obtain a random feasible encoding $r = (r_1, r_2, \ldots, r_N)$ of a feasible schedule.

Compared to the encoding used in pGOMEA from Bosman et al. (2016), our encoding adds an integer part to the encoding keys to capture the shift allocation of an activity. This assignment permutation encoding was also proposed for the multiple machine scheduling problem in Bean (1994).

5.2. Fitness function

The fitness function is used to evaluate solutions during the evolution. A smaller fitness score, or in short score, means a better-performing solution in our context and vice versa. A solution, represented by encoding p, can be evaluated via the fitness function

$$f(p) := w_x D(p) + w_y O(p) + w_z W(p), \tag{13}$$

where D represents the cumulative travel time, O the total shift overtime, and W the total waiting time, respectively ($f(p)$ is equal to the objective function of the MILP in Section 4.2). As explained in Section 5.1, encoding p can be decoded to obtain for each shift the order of its activities; however, to evaluate $f(p)$ also, the planned arrival times at the activities are required (including the base location). These planned arrival times are not encoded and need to be derived.

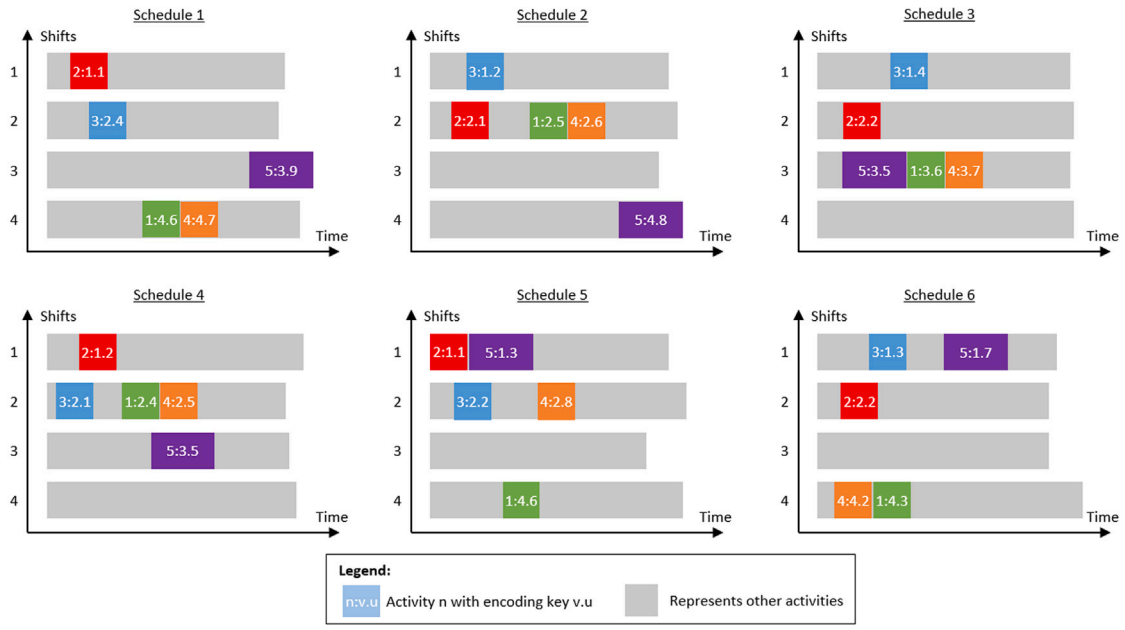


Fig. 2. Running example of an *evolved* (and thus good) population of 6 schedules drawn as Gantt charts in which the focus is on the first five activities that are illustrated and encoded. For example, the encoding of the first five activities in schedule 1 is $(4.6, 1.1, 2.4, 4.7, 3.9)$.

To derive the planned arrival times on a shift when evaluating a solution, we make use of the fact that the order of activities, in combination with the service times, travel times, and start time of the shift, already inductively suggest a planned arrival time at each activity. That is, for an order of activities $(n_1, n_2, \dots, n_k)$, $1 \leq k \leq N$ we can set the arrival time $a_{n_{i+1}}$ at activity n_{i+1} equal to $a_{n_{i+1}} = \max\{s_{n_{i+1}}, a_{n_i} + p_{n_i} + d_{n_i n_{i+1}}\}$ for $i = 0, 1, \dots, k-1$, where we set $a_{n_0} = \max\{0, s_{n_1} - d_{0n_1}\}$. Note that a_{n_0} corresponds to the shift start time; hence, we can also determine a shift start time with this method. Hence, the shift pattern is obtained from the starting times of the shifts with activities assigned to them.

Remark 2. Alternatively, an LP can be solved to find the optimal planned arrival times. Indeed, if the order of arrivals is known then the integer-valued decision variables x_{vnm} (in the MILP formulation in Section 4.2) that represents if activity m directly follows activity n on shift v, is known as well. Hence, x_{vnm} becomes a regular parameter and the MILP is reduced to a standard LP with the arrival times as the main decision variables. In particular, once we remove the constraints in (8) solving the LP leads to optimal arrival times (including shift start times) within polynomial time (Bertsimas and Tsitsiklis, 1997). Nevertheless, solving an LP for each solution we consider in the algorithm would be too time-consuming. It is possible to only solve the LP at certain well-chosen moments or solutions; however, this idea will not be explored in this paper.

5.3. Population

The population $\mathcal{P}$ is a set of solutions, represented by encodings, that is maintained throughout the evolutionary search. Fig. 2 illustrates an example population of 6 schedules. The population size is user-defined and constant, meaning it does not change during the evolutionary search. By only accepting better solutions in the next generation, the performance of the population will not get worse during the evolution. The population is initialized with random feasible solutions (for more details, see Section 5.1). In principle, problem-specific heuristics can be used to initialize the population. However, it is questionable whether that is worth the extra computation effort since, in general (which is also shown later on in our experiments), most progress is made at the first part of the optimization, making it likely to quickly surpass simple heuristically constructed solutions (see the discussion in Section 3.5 of Eiben and Smith (2015)).

5.4. Recombination of solutions

The role of recombination is to create new solutions by combining existing solutions. For a given subset of activities, Section 5.4.1 describes how two solutions are recombined. Section 5.4.2 describes how these subsets of activities can be smartly chosen using a linkage tree to increase the chance of finding better solutions. How to specify the dependency measure to be used when constructing the linkage tree is described in Section 5.4.3.

5.4.1. Recombining two solutions for a given subset of activities

Suppose we want to improve $p \in \mathcal{P}$ by recombining it with another donor solution $d \in \mathcal{P} \setminus p$. We are given a subset of activities $S \subseteq \mathcal{N}$ on which this recombination will be based. To recombine p into a new candidate solution p', replace the keys of encoding p with the keys of encoding d at the indices/activities from S, i.e., $p'_k = d_k$ for $k \in S$, and $p'_k = p_k$ otherwise. If $f(p') < f(p)$, the new candidate solution p' is better than p and thus p is replaced by p' in the population. For example in Fig. 2, if schedule 1 with encoding $(4.6, 1.1, 2.4, 4.7, 3.9)$ is recombined with donor schedule 3 with encoding $(3.6, 2.2, 1.4, 3.7, 3.5)$ on activities $S = \{1, 4, 5\}$, the encoding of candidate schedule becomes $(\underline{3.6}, 1.1, 2.4, \underline{3.7}, \underline{3.5})$ (where the inherited donor keys are underlined), i.e., the new candidate solution schedules (most likely) activity 5 earlier and activities 1 and 4 are rescheduled from shift 4 to shift 3 and are scheduled behind activity 5 (most likely quite near).

5.4.2. Using a linkage tree for subsets of activities

The main idea in paGOMEA is to use activity subsets in the recombination that increase the chance of finding improved solutions. Being able to find improved solutions more easily, one can expect that the optimization becomes more effective.

We want to choose the activity subsets for recombination to maintain important activity dependencies. In particular, the following activity dependencies can be expected to be important (also in permutation assignment problems in general):

1. **Attraction of activities:** Some activities are convenient to be scheduled on the same shift, such as activities 1 and 4 in Fig. 2. So it makes more sense in the example of Fig. 2 to vary activities 1 and 4 together (on different shifts and shift positions) instead of, for example, activities 4 and 5. This dependency may be because the activity locations are near, and the time windows match nicely. In addition, the following dependencies can be relevant on a particular shift (which is again illustrated by activities 1 and 4 in Fig. 2):
   (a) **Proximity of activities:** It is convenient to schedule two activities directly behind each other. For example, the time windows of two activities are such that both should be addressed directly after each other.
   (b) **Order of activities:** It is effective to schedule a specific activity before another activity. For example, when the first activity has a time window that starts earlier than the second time window.
2. **Repulsion of activities:** Some activities should be scheduled on different shifts. For example, because similar time windows and significant activity durations make it impossible to handle multiple activities within their time windows, or because the distance between activities makes it inefficient. This dependency is illustrated in Fig. 2 where activities 2 and 3 seem to alternate between shifts 1 and 2, suggesting that scheduling them together on a shift is inefficient. Therefore, by varying activities 2 and 3 together, there is a larger chance that both remain scheduled on different shifts making it more likely to find a better schedule.

Note that the described activity dependencies are also likely to be relevant in other permutation-assignment problems, such as vehicle routing problems with time windows, for example.

In general, it is impossible to determine upfront what activity dependencies should be maintained for good solutions since it strongly depends on the rest of the schedule and the problem instance (i.e., it cannot be independently determined). For example, two activities may be convenient on the same shift, but doing so may rule out a combination on that particular shift of several other activities that are even more convenient together.

To identify which activity dependencies should be maintained, we make use of the fact that the population as a whole will most likely improve over time (see Section 5.3). If the solutions improve, it is likely that certain important activity dependencies will be exploited. This is illustrated in Fig. 2, where the evolved population suggests that scheduling activities 2 and 3 on different shifts is efficient. Therefore, we will try to identify which activity dependencies are important so that we can increase the chance of maintaining them.

A hierarchical clustering method is used to identify prevalent activity dependencies in the population, meaning that during the evolution, when the population becomes better, it is likely that important activity dependencies are identified. In particular, the unweighted pair group method with arithmetic mean (UPGMA, see Müllner (2011)) is used to find a linkage tree that reflects the structure present in a given pairwise dependency matrix. This linkage tree constitutes a family of subsets (FOS), a collection of subsets of activities with a relatively large dependency as measured by a given dependency measure. The idea is to define a dependency measure that scores all pairwise activities in terms of the previously mentioned activity dependencies.

Before elaborating on an effective dependency measure in the next subsection, let us first specify how the linkage tree construction constitutes a FOS for a given dependency measure $\delta(n,m)$ for all pairwise activities $n,m \in \mathcal{N}$. An FOS, denoted by $\mathcal{F}$, is a collection of subsets on the set of activities $\mathcal{N}$, i.e., $\mathcal{F} = \{F^0, F^1, \ldots, F^{|\mathcal{F}|-1}\}$, where $F^i \subseteq \mathcal{N}$. The sets F^i are called linkage sets. These linkage sets represent activities that have a large dependency. The linkage tree, and thus the FOS, is built in the following way. Initialize $\mathcal{F}$ with N singleton linkage sets $F^i = \{i\}$, $i \in \mathcal{N}$. Search within $\mathcal{F}$ for the two linkage sets F^i and F^j, $i \neq j$, which have on average the highest dependency between the activities contained in the linkage sets. Include the new linkage set $F^i \cup F^j$ in the FOS $\mathcal{F}$; the linkage sets F^i and F^j stay in the FOS but are not considered for merging anymore. This process is repeated and completed once a linkage set including all activities is added to the FOS. Fig. 3 illustrates the FOS for the running example from Fig. 2. Note that $|\mathcal{F}| = 2N - 1$.

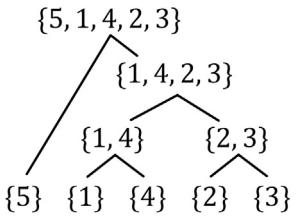


Fig. 3. Linkage tree for the running example from Fig. 2 (with 5 activities). All 9 linkage sets form the FOS to be used in paGOMEA.

5.4.3. Dependency measure

Choosing an effective dependency measure allows the linkage tree to identify effective activity subsets for recombination. The dependency measure $\delta(n,m)$ should be relatively large if the joint recombination of activities n and m increases the chance of finding better solutions, and small otherwise. To this end, for each activity pair, the dependency measure scores the amount of attraction and repulsion as specified earlier. As a result, it is more likely that the recombination maintains activities that are attracted to each other and, similarly, separates activities that repel each other.

The dependency measure is determined by considering the population of schedules and looking at the position of the two activities for each schedule. Thus, the purpose of the dependency measure is to infer from the population if, and to what extent, two activities are related in light of the earlier defined activity dependencies. To measure the dependency between two activities, we proceed in two steps: (1) we investigate if the two activities attract or repulse each other in some sense, i.e., if the activities are frequently assigned to the same shift or to different shifts; (2) we investigate how the activities are related within the same shift in the case of attraction, and between the different shifts in the case of repulsion. The relation of two activities within the same shift and between different shifts will be referred to as the relation on an internal and external level, respectively. Note that observations on attraction and repulsion, as well as observations on the external level, have no meaning if there is just one available shift (i.e., $V = 1$). In that case, it makes sense to define the dependency measure only on the internal level as in Bosman et al. (2016).

To define δ we first consider a dependency measure that quantifies the level of attraction or repulsion; after this, we will define dependency measures that operate on an internal and external level. Finally, we combine these dependency measures to obtain δ. In what follows, a strong dependency between two activities is reflected in the dependency measure with a high value, and a weak dependency is reflected with a low value.

Attraction and repulsion. Let us focus on attraction, as repulsion follows from a similar approach. The dependency measure for attraction of two activities n and m is based on the number of times activities n and m are scheduled on the same shift counted over all schedules in the population. Therefore, denote x_{nm} as the number of same-shift schedules in the population for activities n and m. To infer the level of attraction from the observation of x_{nm}, we compare the observed value with the value that we expect to observe in a random population, i.e., when for each schedule in the population all activities are randomly assigned to a feasible shift. To this end, define the random variable X_{nm} for

the (hypothetical) number of same-shift schedules for activities n and m, given that we are observing a random population instead. For a population of size $P \in \mathbb{N}$ the random variable X_{nm} is modeled as a binomial distributed random variable with P being the number of trials and with a success probability given by

$$\pi_{nm} := \frac{|C_n \cap C_m|}{|C_n||C_m|},$$

with C_n, C_m the set of feasible shifts for activities n and m, respectively. Now, if x_{nm} far exceeds the expected value $\mathbb{E}X_{nm} = P\pi_{nm}$ we conclude that the number of same-shift schedules in the population is not due to randomness but due to some level of attraction. To quantify the level of attraction, we adopt the statistical view that the more unlikely it is for x_{nm} to be observed in a random population, the greater the level of attraction. On the contrary, we similarly measure the level of repulsion if x_{nm} is much smaller than $P\pi_{nm}$. In view of this, the dependency measure for attraction and repulsion is defined as

$$\delta^{\text{stat}}(n,m) := 1 - \begin{cases} \mathbb{P}(X_{nm} > x_{nm})/\mathbb{P}(X_{nm} > P\pi_{nm}), & \text{for } x_{nm} > P\pi_{nm}, \\ \mathbb{P}(X_{nm} \leq x_{nm})/\mathbb{P}(X_{nm} \leq P\pi_{nm}), & \text{for } x_{nm} \leq P\pi_{nm}, \end{cases} \tag{14}$$

where we have normalized the dependency measure to attain values in $[0,1]$.

Internal level. On the internal level, we consider the dependency between two activities when scheduled on the same shift. The dependency involves the ordering of the activities and the proximity of the activities, respectively. To capture these internal dependencies in terms of the population encodings, define the population in terms of its encodings by $\mathcal{P} := \{(r_1^k, r_2^k \dots, r_N^k) : k = 1,2,\dots,P\}$, and let $I_{nm} := \{k : \lfloor r_n^k \rfloor = \lfloor r_m^k \rfloor \text{ for } k = 1,2,\dots,P\}$ be all population members with activity n and m scheduled on the same shift. Assume that $|I_{nm}| > 0$ for the time being.

The so-called relative-ordering information measures to what extent there is a pattern in the order of two activities n and m when scheduled on the same shift. It is defined as

$$\delta_1^{\text{int}}(n,m) := 1 - \{-[\hat{p}_{nm}\log_2(\hat{p}_{nm}) + (1-\hat{p}_{nm})\log_2(1-\hat{p}_{nm})]\}, \tag{15}$$

where

$$\hat{p}_{nm} := \frac{1}{|I_{nm}|}\sum_{k \in I_{nm}} \mathbb{1}_{\{r_n^k < r_m^k\}}$$

is a population-based probability of finding n to be scheduled before m when assigned to the same shift. Relative-ordering information $\delta_1^{\text{int}}(n,m) \in [0,1]$ is an inverted entropy that is relatively large when n is often scheduled either before or after m in the population. When there is no clear pattern regarding the order of n and m in the population, $\delta_1^{\text{int}}(n,m)$ will be small. As an example, $\delta_1^{\text{int}}(1,4)$ will be relatively large in Fig. 2 as activity 1 is often scheduled before activity 4 when assigned to the same shift.

The adjacency information measures the distance between two activities when scheduled on the same shift. For activities n and m, it is defined as

$$\delta_2^{\text{int}}(n,m) := 1 - \frac{1}{|I_{nm}|}\sum_{k \in I_{nm}} (r_n^k - r_m^k)^2. \tag{16}$$

Adjacency information $\delta_2^{\text{int}}(n,m) \in [0,1]$ will be large when the key difference between n and m is small, i.e., it is likely that n is scheduled near m. Vice versa, $\delta_2^{\text{int}}(n,m)$ is small when it is unlikely that n is scheduled near m. As an example, $\delta_1^{\text{int}}(1,4)$ will be relatively large in Fig. 2 since activity 1 is often scheduled near activity 4 when assigned to the same shift. In case that $|I_{nm}| = 0$ we set both $\delta_1^{\text{int}}(n,m)$ and $\delta_2^{\text{int}}(n,m)$ equal to 0.

The relative-ordering information and the adjacency information are combined into the following dependency measure on the internal level

$$\delta^{\text{int}}(n,m) := \delta_1^{\text{int}}(n,m)\delta_2^{\text{int}}(n,m). \tag{17}$$

Compared to the dependency measure from Bosman et al. (2016), $\delta^{\text{int}}(n,m)$ is only adjusted to account for the same shift.

External level. On the external level, we consider the dependency between activities n and m when they are positioned on different shifts. The main question is to what extent activities n and m are related based on their shift assignment. More specifically, if we observe that activity n is assigned to some shift, how much information does this give about the shift assignment of activity m? For example, it could be that two shifts are both efficient for two activities (maybe because of suitable start and end times of the shifts), but the time windows of the two activities collide, so scheduling both activities on one of the two shifts is inefficient. Naturally, in the population of schedules, this could lead to the two activities alternating between the two shifts, such as activities 2 and 3 in Fig. 2. The measure that we employ to find such dependencies between activities is the so-called mutual information (Cover, 1999).

To introduce mutual information define two random variables S_n and S_m such that $S_n = v$ if activity n is assigned to shift v and $S_m = w$ if activity m is assigned to shift w, where $v \in C_n$ and $w \in C_m$. Furthermore, define the joint probability distribution,

$$q_{nm}(v,w) := \mathbb{P}(S_n = v, S_m = w),$$

and denote $q_n(v) := \mathbb{P}(S_n = v)$ and $q_m(w) := \mathbb{P}(S_m = w)$ for the marginal distributions. The entropy measure of mutual information is defined as

$$I(S_n, S_m) := \sum_{v \in C_n, w \in C_m} q_{nm}(v,w)\log_{c_{nm}}\left(\frac{q_{nm}(v,w)}{q_n(v)q_m(w)}\right),$$

with $c_{nm} := \min\{|C_n|, |C_m|\}$ (the choice for c_{nm} assures that the mutual information measure is bounded by 1). Note that entropy is a measure that captures the uncertainty in a random variable. Mutual information is the reduction in the uncertainty of one random variable due to the knowledge of another random variable. In this sense, $I(\cdot,\cdot)$ is a measure of dependency between the two random variables S_n and S_m. It is symmetric in n and m and equal to 0 if S_n and S_m are independent. It can be shown that $0 \leq I(S_n, S_m) \leq H(S_n), H(S_m) \leq 1$, with $H(\cdot)$ the entropy of a random variable. The dependency measure that we use is the empirical version of mutual information:

$$\delta^{\text{ext}}(n,m) := \sum_{v \in C_n, w \in C_m} \hat{q}_{nm}(v,w)\log_{c_{nm}}\left(\frac{\hat{q}_{nm}(v,w)}{\hat{q}_n(v)\hat{q}_m(w)}\right), \tag{18}$$

where $\hat{q}_{nm}(v,w), \hat{q}_n(v)$ and $\hat{q}_m(w)$ are the corresponding empirical probabilities based on the population, e.g., $\hat{q}_n(v)$ is the relative number of times that activity n is scheduled on shift v. Measure $\delta^{\text{ext}}(n,m)$ will be relatively large when it is unlikely that activities are often assigned to different shifts just by chance (which suggests that n and m repel each other for some reason), and it is small otherwise. As an example, $\delta_1^{\text{ext}}(2,3)$ will be relatively large in Fig. 2 since activities 2 and 3 seem to repel each other on the first two shifts.

Combining the dependency measures. To define the full dependency measure δ, first consider the dependency measure δ^{stat} of attraction and repulsion defined in Eq. (14) as it plays a key role in its definition. It is clear that δ^{stat} distinguishes between two complementary situations: a surmount of same- or different-shift schedules in the population. In the first case the dependency measure should be considered on an internal level with δ^{int} defined in Eq. (17), and in the second case on an external level with δ^{ext} defined in Eq. (18). Moreover, δ^{stat} quantifies the extent of attraction or repulsion in these cases; hence it should regulate the intensity with which we consider the internal and external levels. Finally, δ^{stat} acts as a dependency measure in itself. Based on these arguments, we define the full dependency measure as follows

$$\delta(n,m) = \begin{cases} \delta^{\text{stat}}(n,m)(w + (1-w)\delta^{\text{int}}(n,m)), & \text{for } x_{nm} > P\pi_{nm}, \\ \delta^{\text{stat}}(n,m)(w + (1-w)\delta^{\text{ext}}(n,m)), & \text{for } x_{nm} \leq P\pi_{nm}, \end{cases} \tag{19}$$

with weight $w \in [0,1]$ that assigns the relative importance of the level of attraction or repulsion compared to the dependency measure on the

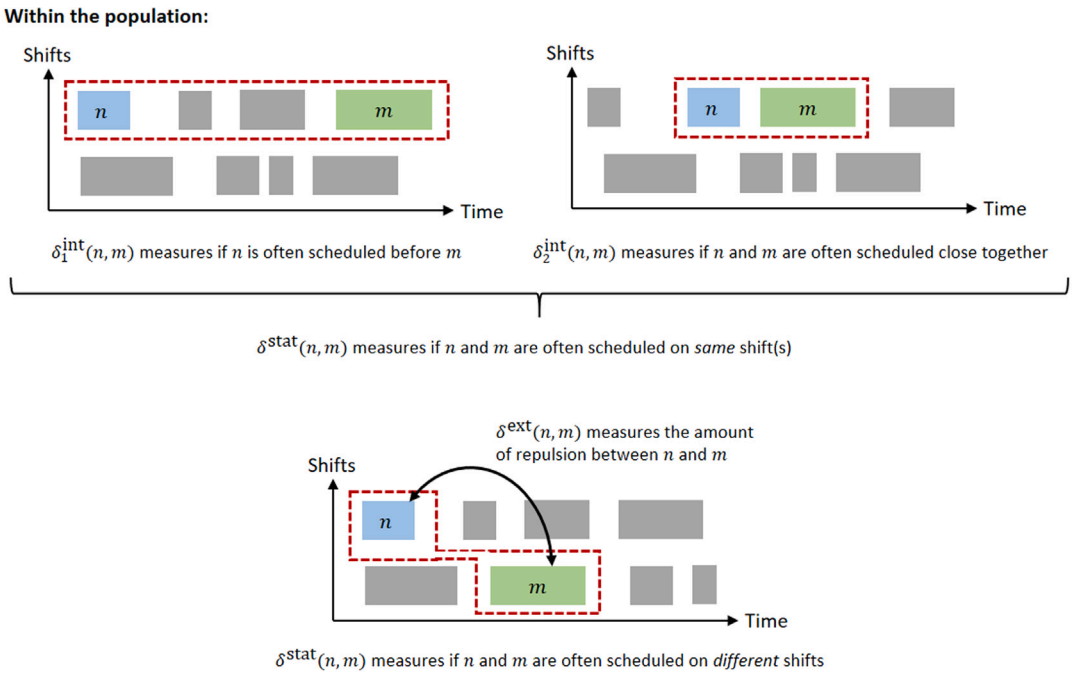


Fig. 4. Components of the dependency measure.

internal or external level, respectively. See Fig. 4 for a visual illustration and (intuitive) summary of the components of the dependency measure.

For illustration purposes, the linkage tree from Fig. 3 is based on δ calculated for the example from Fig. 2. Indeed, it shows that activities 1 and 4 will be recombined (because of their attraction), and similarly, activities 2 and 3 will be recombined (because of their repulsion). As a result, the chance of a successful recombination increases. For example, because recombination of activities 2 and 3 is stimulated in Fig. 2, there is a larger chance that both remain scheduled on different shifts.

Remark 3. We also experimented with other candidate dependency measures, as it is not always a priori clear which measure performs best. For example, as δ^{int} and δ^{ext} are entropy-based dependency measures, an alternative choice for δ^{stat} is to use an (inverted) asymmetric entropy measure (Guermazi et al., 2018), which has similar characteristics as δ^{stat}. Based on our numerical experiments, the performance of δ^{stat} was superior to such alternative measures.

5.5. Evolution of the population

In each generation, paGOMEA tries to improve every solution p from the current population $\mathcal{P}$. For each solution p, paGOMEA randomly loops over all subsets from the current FOS. For each subset, a random donor solution $d \in \mathcal{P} \setminus p$ is picked and used for the recombination based on the current subset of activities. This leads to a new candidate solution p' that is evaluated. Note that a new candidate solution p' is always feasible because the initial population contains only feasible solutions. The candidate solution only replaces p in the population when it is better. This procedure is repeated for every solution in $\mathcal{P}$. Afterward, the above procedure repeats on the population of the new generation until a termination condition is met.

At the start of each generation, the population is re-encoded to stimulate diversity to prevent early stagnation of the optimization. To that end, each individual is re-encoded by randomly generating keys such that decoding of those keys gives the same schedule back again. More specifically, the integer parts of the keys remain the same, but the decimal parts are replaced by random numbers from $(0, 1)$. If a shift contains A activities, A random numbers from $(0, 1)$ are drawn, sorted from small to large, and re-assigned to the activities in order of the original schedule on that shift. Directly after the population has been re-encoded at the start of each generation, the FOS is re-determined based on the current population.

5.6. Termination condition

The procedure of paGOMEA terminates when some stopping condition is reached, e.g., once a given number of generations has been reached or when the improvement becomes relatively low for a certain number of consecutive generations. The final outcome is the best solution in the final generation with the smallest fitness function f value.

5.7. Procedure of paGOMEA

The pseudo-code of paGOMEA, which uses all previously discussed elements, is given in Algorithm 1. By recombining solutions based on the linkage tree based on an evolving population, paGOMEA promotes solution variations that respect:

- Shift preferences of activities.
- Relative positions of activities on a shift (for example, early or late on a day).
- Sets of activities that work well together on a shift, especially when scheduled in a certain order.
- Sets of activities that are inefficient to be scheduled on the same shift.

Therefore, compared to traditional EAs, it is more likely that improved solutions are found during recombinations, and thus one can expect that the optimization becomes more effective.

A downside, however, is that using a linkage tree to guide the evolutionary search increases the running time to construct the FOS at the start of each generation. Given that paGOMEA runs for G generations, the running time of paGOMEA is $\mathcal{O}(GPN^2)$. In particular, in each generation, first the FOS has to be constructed in running time $\mathcal{O}(PN^2)$, i.e., $\mathcal{O}(PN^2)$ to calculate the dependency measures and $\mathcal{O}(N^2)$ to construct the linkage tree (Müllner, 2011). Afterward, each

Algorithm 1 paGOMEA

```
P ← random start population of encoded solutions
repeat
    re-encode all solutions in P
    FOS ← build from linkage tree based on current population
    for solution p ∈ P do
        for linkage-set F ∈ FOS (in random order) do
            d ← choose a random donor solution from P
            p′ ← replace keys of p by d for activities from F
            if f(p′) < f(p) then
                p ← p′    (p′ replaces p in P)
            else
                continue
until stopping condition
return argmin{f(p) : p ∈ P}
```

solution gets recombined $2N-1$ times into a candidate solution that is evaluated in $O(N)$, which thus runs in $\mathcal{O}(PN^2)$ as well. Our numerical experiments in the next section demonstrate that more effective recombinations are worth the extra computational effort (see the analysis of Fig. 6).

A possible issue of using a linkage tree to guide the evolutionary search is that the diversity of the population may drop earlier during the evolution leading to stagnation in the optimization (especially when the population size is small or the number of generations is large). In that case, one may restart paGOMEA with a new random start population, possibly including the previously best-found solution, to promote further exploration of the solution space. Alternatively, one could introduce a mutation operator to promote exploration. In the numerical experiments presented in the next section, the number of generations and population sizes were such that it was not an issue.

6. Numerical results

In this section, we benchmark the performance of paGOMEA against schedules that result from various other scheduling methods. Furthermore, we carry out a sensitivity analysis to investigate the impact of different weights for the score components in the fitness function from (13) (i.e., w_x for the travel time, w_y for the shift overtime and w_z for the waiting time).

The instances used for the experiments are based on three real-life cases of a Dutch HHC organization. Here we distinguish between the experiments performed on the larger sets of unaltered real-life data directly obtained from the HHC organization, and experiments performed on smaller sets of real-life inspired data. The real-life inspired data sets are created so that the experiments can be performed on data sets of various sizes and complexity; the data comprises a mix of bootstrapped data from the real-life data sets and data that we generated ourselves (see Section 6.4 for more details).

After introducing the three cases in Section 6.1, we explore the case study instances to gain more insight into how care activities and shifts are distributed across the day. Next, in Section 6.2, these instances are used to compare the different algorithms that result from choosing different versions of the dependency measure. In particular, we compare paGOMEA to (a slightly modified) pGOMEA from Bosman et al. (2016) and to an algorithm comparable to a standard evolutionary algorithm. In addition, the results of paGOMEA are compared to human-made schedules from the HHC organization in Section 6.3. In Section 6.4, we consider the real-life inspired data to benchmark paGOMEA to a MILP solver (Gurobi version 9.0.2) that produces optimal solutions for small instances in reasonable time. This real-life inspired data is also used in Section 6.5 to perform a sensitivity analysis with respect to the fitness function weights.

Table 2
Summary of model parameters from city region, sub-urban region and rural region; time unit is minutes.

	City	Sub-urban	Rural
Nr. of activities	87	71	90
Nr. of shifts (qual. level)	4 (2), 3 (3)	7 (2), 3 (3)	5 (2), 6 (3)
Avg. travel time (std.)	1.27 (0.85)	2.07 (1.42)	5.36 (3.7)
Avg. service time (std.)	17.13 (10.16)	22.46 (13.21)	20.44 (11.78)
Avg. shift duration (std.)	317.14 (27.11)	219.50 (43.21)	245.46 (13.22)

6.1. Case study: data analysis

For the numerical experiments, we use real-life data from three different HHC teams of a Dutch HHC organization. Each team operates in a separate geographical region, and the regions that we consider are a city region, a sub-urban region and a rural region. The model parameters that result from the three underlying data sets are summarized in Table 2 and have the following assumptions and characteristics:

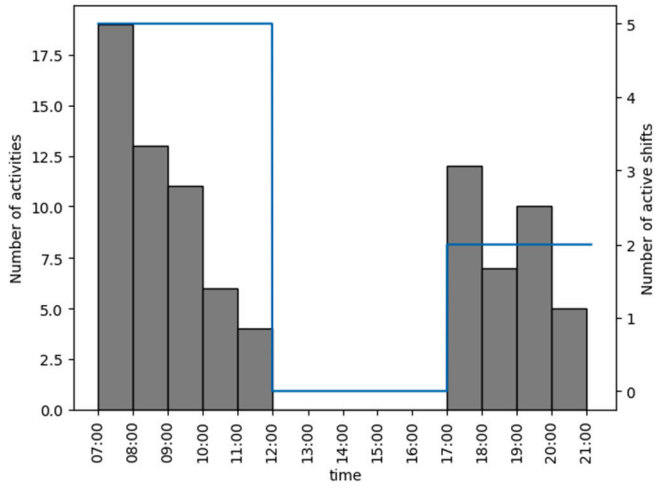
- The number of activities, the required qualification level for each activity and the travel time (by car) between two activities are given.
- The number of available shifts, qualification level of each shift and the shift durations are given.
- The time windows of the activities for each of the schedules were for a large part absent; in case no time window was given we made an estimation based on the arrival times in the real schedule. The time windows were set to be one hour long by setting the start and due time half an hour before and after the arrival time, respectively.

From Table 2 we observe that the average travel time between each pair of activities, as expected, increases as we move from the city region to the rural region. It is interesting to note that there are only two qualification levels: a qualification level for non-medical care activities (washing, dressing, etc.) and non-complex medical care activities (drug administration, treatment of wounds, etc.). Care activities that involve complex treatment appear relatively infrequent and are often assigned to one special shift with the required qualification level, rather than included in the regular schedule. There does not seem to be a distinguishing pattern for the service time or shift duration per region. Finally, Fig. 5 provides insight into the number of scheduled activities throughout the day, as well as the current distribution of shifts. As can be observed, there is a peak of activities in both morning and evening, with (almost) no scheduled activities in between.

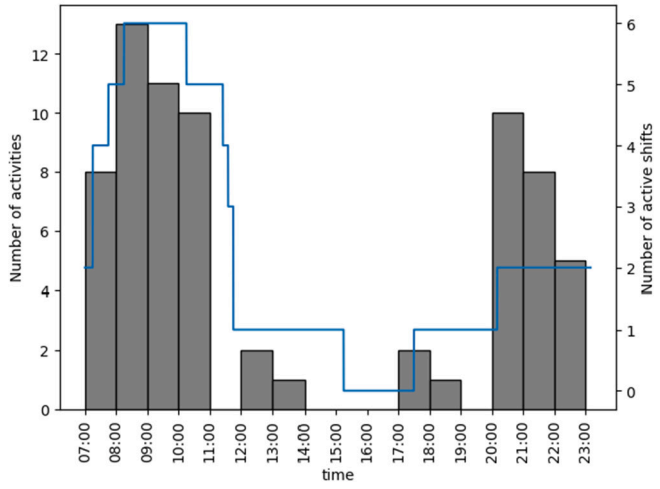
6.2. Comparison of algorithms

We will now investigate the performance of the algorithm by comparing the effect of different choices in terms of types of dependency measures and population sizes. Recall that the dependency measure determines what kind of linkage tree is learned and used to guide the evolutionary search of paGOMEA (see Section 5). We consider the following three types of dependency measures:

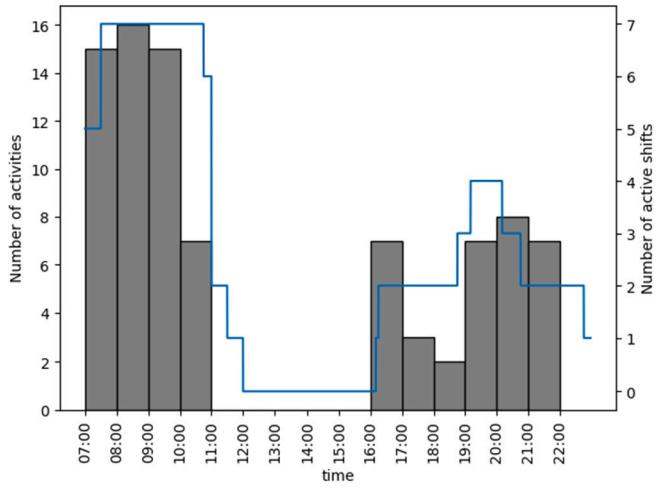
1. Full dependency measure δ as given in Eq. (19).
2. Internal dependency measure δ^{int} as given in Eq. (17), which is a slight modification of the dependency measure used for the (standard) pGOMEA in Bosman et al. (2016) (see the discussion in Section 5).
3. Random dependency measure δ^{rand}, where $\delta^{\text{rand}}(i,j)$ is an independent random sample from a uniform distribution on $(0,1)$ for all $i \neq j$. A random dependency measure gives no structure to the linkage tree; hence the activities that are chosen for recombination become completely random. Therefore, the algorithm with a random dependency measure is comparable to a standard evolutionary algorithm that does not learn and exploit a model to guide the evolutionary search.



(a) City region.



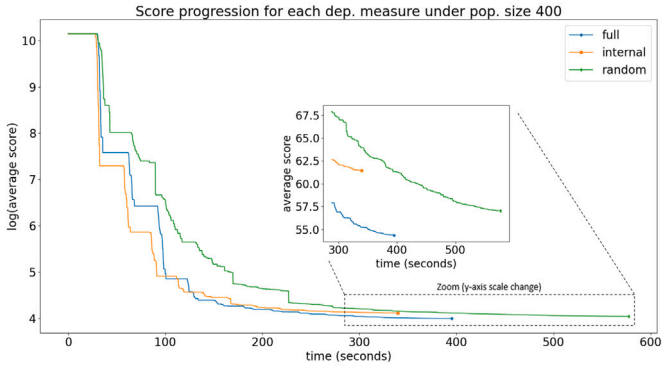
(b) Sub-urban region.



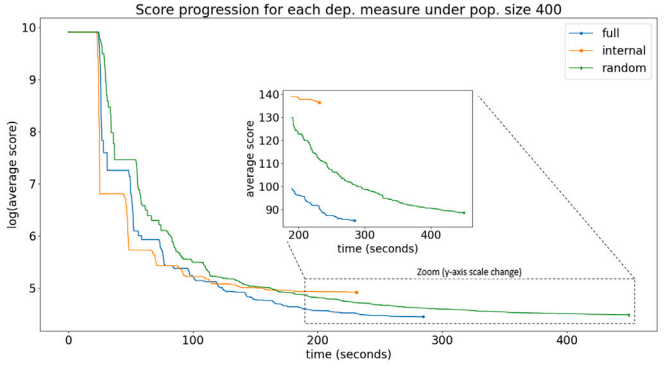
(c) Rural region.

Fig. 5. Distribution of scheduled activities and shifts throughout the day per region.

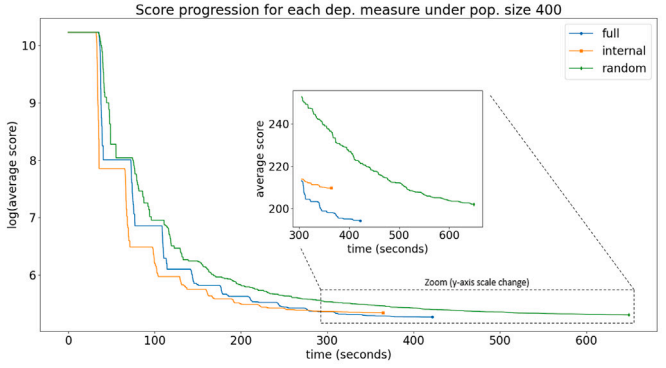
For each of the instances, and in turn, for each of the dependency measures we proceed by running the algorithm R times for selected



(a) City region.



(b) Sub-urban region.



(c) Rural region.

Fig. 6. Average score progressions over time for $R = 30$ iterations for population size 400 under measures δ (full), δ^{int} (internal) and δ^{rand} (random) for the city, sub-urban and rural region instances.

start populations of size P, where we set the weights of the fitness function (13) $w_x = w_y = w_z = 1$ and the weight in δ to $w = 2/3$. Hence we obtain a result for each quadruple (instance, dependency measure, population size, iteration). The result consists of the progression of best scores throughout the generations. If the relative improvement of the score progression per generation is less than 1% for two consecutive generations, we stop the process. We set the maximum number of generations equal to 20. Furthermore, the computation time for each generation cycle is recorded; this way, we can obtain the current best score at a selected time. For a better comparison, we select the same (random) start population for all dependency measures if the instance, population size and iteration are equal.

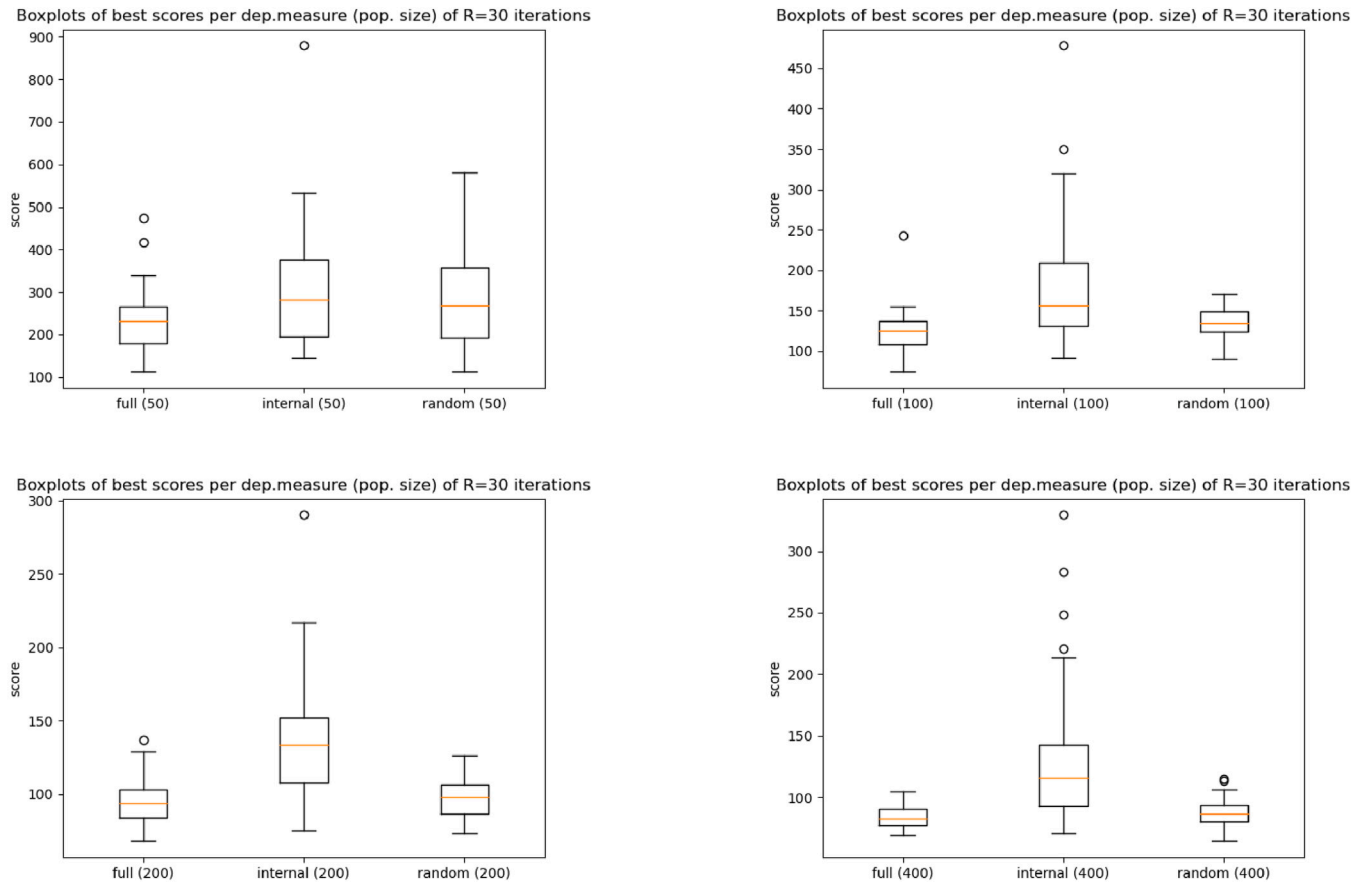


Fig. 7. Boxplots on the best score in $R = 30$ final generations for δ (full), δ^{int} (internal) and δ^{rand} (random) over various population sizes for the sub-urban region instance.

In Fig. 6 the average score progressions over time are displayed on a logarithmic scale for $R = 30$ and $P = 400$ over all three case study instances for the measures δ, δ^{int} and δ^{rand}. Furthermore, we zoomed in at the moment of convergence for all three dependency measures using a logarithmic scale to better compare the final scores. It stands out that on average the algorithm under δ^{int} converges faster to a final solution compared to the algorithm under δ and δ^{rand}. However, by the time the algorithm under δ^{int} is stopped, the algorithm under δ already reached a better score. This is not the case for δ^{rand}, but the algorithm under δ^{rand} on average outperforms the algorithm under δ^{int} when comparing the final solutions. Finally, the algorithm under δ on average outperforms the algorithm under δ^{rand} both in time and score. Also note that it reaches on average a better solution at each point in time. For the other population sizes $P = 50, 100, 200$ we reach a similar conclusion based on the same type of figures in the Appendix with an exception for $P = 100$ of the city region instance (see also the discussion of the results in Table 3).

Fig. 7 displays a box plot of the best score per final generation for each $R = 30$ runs for various population sizes per measure δ, δ^{int} and δ^{rand} for the sub-urban region, and provides insight into the spread of the scores. In these experiments the variety in scores is typically somewhat smaller under δ and δ^{rand} than under δ^{int}, while there does not seem to be much difference in the variety when comparing the algorithm under δ and δ^{rand}. The boxplots also show that as the population size increases it is more probable to reach a better score under a given dependency measure (however, this comes with a trade-off with respect to run time, see Table 3). More or less the same conclusion can be drawn for the city and rural region instance from the corresponding figures in the Appendix.

Finally, Table 3 displays all averages and 95%-confidence intervals corresponding to the data used for the boxplots, as well as the average run time, for all case study instances. For the most part the results confirm the conclusions based on Figs. 6 and 7. The only exception is for the city region instance when $P = 100$. Only in this case, the algorithm under δ^{rand} outperforms δ with respect to average score (which does not hold for larger population sizes). Although the median score for δ (64.44) is still smaller than the median score for δ^{rand} (65.92) (see also the corresponding boxplots in the Appendix), there are some experiment runs where paGOMEA finds and exploits highly inefficient substructures. As a result, the algorithm under δ sometimes gets stuck in a poor local optimum which brings down the average performance as compared to the algorithm under δ^{rand}. We observe that this problem does not occur for larger population sizes. This is most likely because a larger population has a larger variety of solutions, hence more exploration takes place.

6.3. Case study: numerical results

For the comparison of schedules between paGOMEA and the case study schedules created by human-schedulers we compute the scores for the full case study instances with paGOMEA in an identical manner as in Section 6.2. In particular, we run paGOMEA under the full dependency measure δ and population size of $P = 400$, and do it only once (i.e., $R = 1$) per instance. The results in Table 4 show that paGOMEA can significantly improve the schedules from the case study. As the time windows were absent for the city and rural regions, the waiting times for the benchmark are zero by construction. It is likely that the preferences of clients are assumed by caregivers or unknown for these two regions, providing room for improvement regarding client-centered care. For the sub-urban region, paGOMEA almost fully eliminates any waiting of clients. Moreover, paGOMEA manages to avoid any shift overtime. Maybe the most striking is that travel times can be reduced

Table 3
Averages and normal 95% confidence intervals on the best score in $R = 30$ final generations for δ, δ^{int} and δ^{rand} over various population sizes and regions; run time average in seconds is denoted inside brackets next to the confidence interval.

δ	$P = 50$	$P = 100$	$P = 200$	$P = 400$
City	471.31 ± 76.14 (46.9 s)	183.96 ± 56.45 (90.7 s)	56.63 ± 0.64 (208.09 s)	54.12 ± 0.73 (395.36 s)
Sub-urban	232.72 ± 27.7 (32.18 s)	125.53 ± 10.64 (71.52 s)	96.09 ± 6.07 (140.18 s)	85.13 ± 3.3 (284.67 s)
Rural	625.99 ± 69.3 (49.63 s)	267.65 ± 26.62 (112.09 s)	220.03 ± 8.12 (211.75 s)	193.82 ± 3.82 (421.73 s)
δ^{int}	$P = 50$	$P = 100$	$P = 200$	$P = 400$
City	640.66 ± 92.66 (31.02 s)	458.4 ± 72.68 (61.02 s)	179.39 ± 55.83 (154.22 s)	61.0 ± 7.16 (339.59 s)
Sub-urban	309.47 ± 52.64 (24.46 s)	185.09 ± 29.38 (53.64 s)	137.41 ± 14.98 (111.12 s)	136.06 ± 22.0 (231.04 s)
Rural	771.22 ± 96.63 (38.57 s)	376.83 ± 54.56 (77.71 s)	245.78 ± 19.33 (175.17 s)	208.82 ± 5.36 (364.36 s)
δ^{rand}	$P = 50$	$P = 100$	$P = 200$	$P = 400$
City	554.3 ± 75.21 (40.11 s)	119.08 ± 47.02 (117.85 s)	59.24 ± 1.08 (272.02 s)	56.2 ± 0.86 (577.63 s)
Sub-urban	282.09 ± 41.02 (38.31 s)	135.32 ± 7.15 (94.7 s)	97.79 ± 4.87 (205.68 s)	87.86 ± 4.19 (449.61 s)
Rural	765.67 ± 99.55 (54.52 s)	336.6 ± 41.7 (127.19 s)	221.04 ± 5.27 (372.93 s)	200.9 ± 3.26 (649.27 s)

Table 4
Comparison of the overall score and for each component: case study benchmark (left) compared to result paGOMEA (right) and relative improvement in brackets; time unit is minutes.

	Score		Travel time		Waiting time		Shift overtime	
City	84.07	50.06 (41%)	84.07	50.06 (41%)	0	0 (0%)	0	0 (0%)
Sub-urban	243.44	69.42 (72%)	101.00	67.43 (33%)	95.48	1.99 (98%)	46.97	0 (100%)
Rural	305.87	172.88 (44%)	282.55	172.61 (39%)	0	0.27 (0%)	23.31	0 (100%)

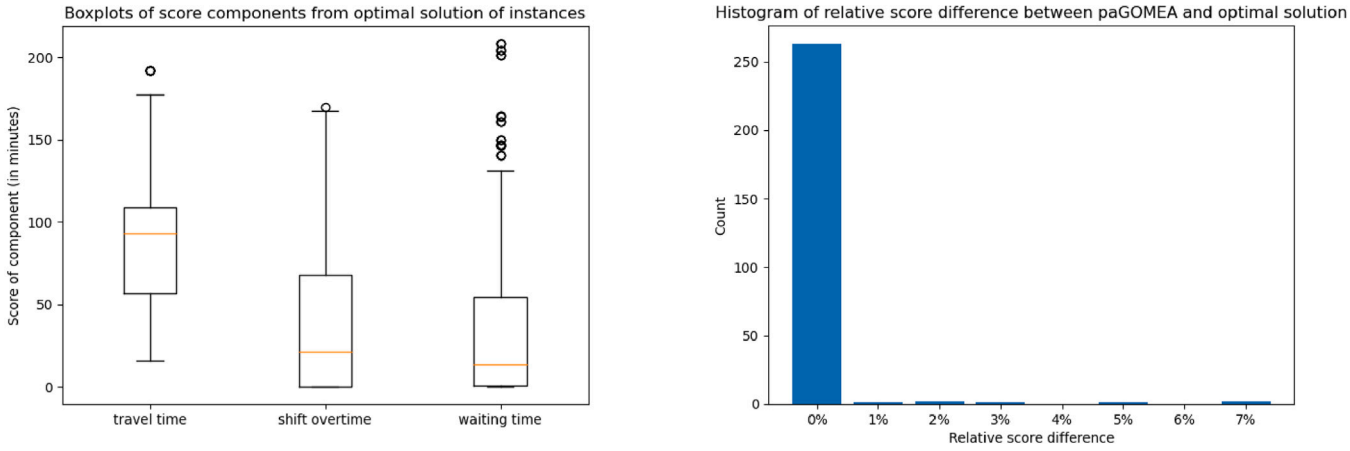


Fig. 8. Relative difference in score of paGOMEA and optimal solution counted over all 270 instances (right), and score component boxplots of optimal solution over all 270 instances.

by at least 33% (for the sub-urban region), and even more for the other two regions. The overall improvements are largest for the sub-urban region due to improvement potential in terms of waiting and shift overtime. Overall, this case study reveals the potential of applying paGOMEA in practice.

6.4. MILP benchmark

To compare the results of paGOMEA and the MILP solver, we created a wide variety of small-sized instances inspired by the case study instances. To do so, we generated for each region (i.e., city, sub-urban and rural) a set of 10 'base' instances that we later extended to complete instances.

Each base instance was created by bootstrapping, and for each activity on the instance, a service time, qualification level, and location from the data set corresponding to the given region. For the locations, we added a small perturbation and scaled them so that the average travel time between pairs of activities varied between instances. Furthermore, we determined for each activity a random start time of the time window and set the number of (4-h) shifts to the least amount necessary to cover the total work.

Each base instance is extended to a complete instance by determining a fixed time window length for each activity and by reducing the base shift durations of 4 h by a fixed amount of time. We tried out different configurations for the time window lengths and reduced shift durations to vary the complexity. The configurations consist of all combinations one could make from 15, 30, or 60 min time window lengths and 180, 210, or 240 min shift durations (note that no time is reduced for shift durations of 240 min), creating in total 9 different configurations (e.g., one configuration could be a time window length of 15 min and a reduced shift duration of 210 min). To clarify, two different instances that stem from the same base instance have the same set of activities and shifts but different time window lengths and shift durations. Hence, for all 3 regions, we ended up with 90 complete instances which stem from 10 base instances, creating 270 instances in total. All instances contain a total number of 12 activities.

For each instance, we computed the optimal schedule with regard to the objective function in Eq. (1) with weights $w_x = w_y = w_z = 1$, by using the MILP solver Gurobi (version 9.0.2) to solve the MILP from Section 4. The resulting score of the objective function is used as a benchmark. To get an idea of the 'variety' of the instances, a boxplot for each of the score components (i.e., travel time, shift overtime, and waiting time deduced from the optimal schedule) is presented in Fig. 8 (left). To compute for each instance the score (of the same objective) with paGOMEA, we proceed with the same settings as described in Section 6.3. For each of the 270 instances, we compared the score of

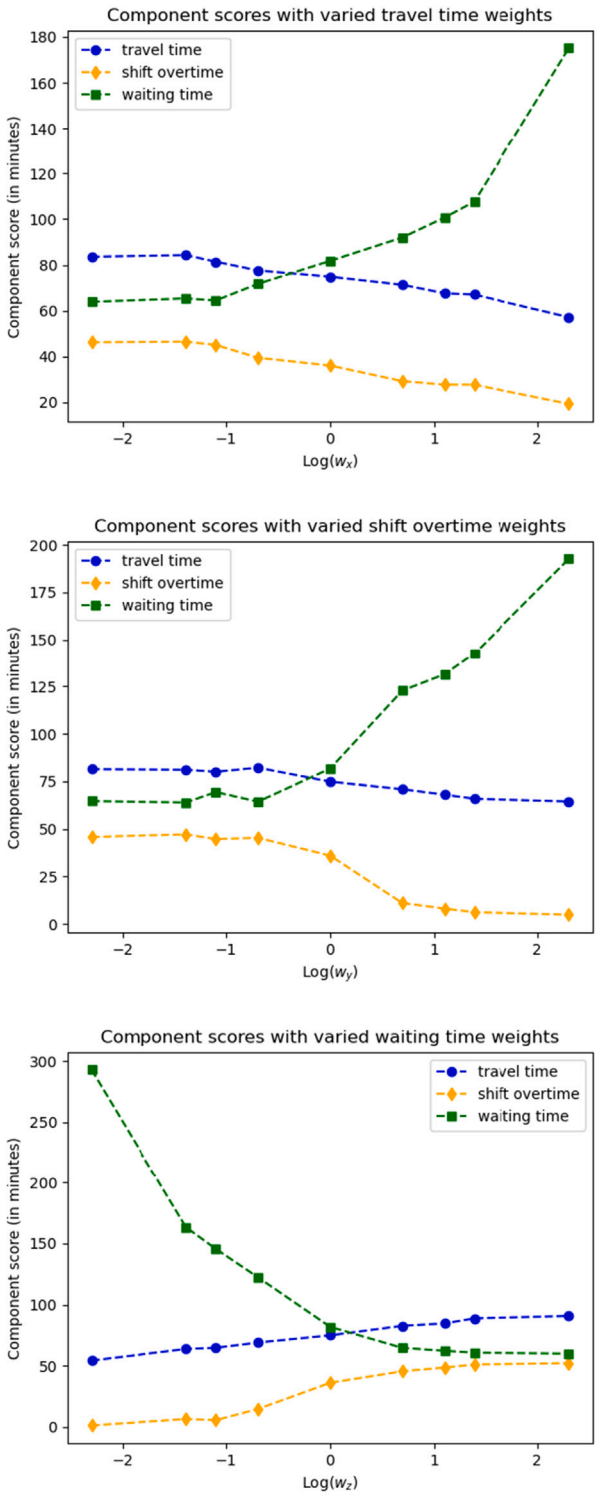


Fig. 9. Average component score with weight for selected components $w_x, w_y, w_z \in \{1/10, 1/4, 1/3, 1/2, 1, 2, 3, 4, 10\}$ (under the natural logarithm), respectively, while fixing the weights of other components to 1.

the optimal solution to the score of the solution computed by paGOMEA and calculated the relative difference in score between paGOMEA and the optimum. The results are presented in Fig. 8 (right). It turns out that paGOMEA performs close to optimal for small-sized instances. Although it is not sure that this difference remains small when instance sizes grow (regarding the number of activities/shifts), it does give confidence that paGOMEA is capable of finding high-quality solutions for larger instances. Finally, we mention that the average run time of paGOMEA over all 270 instances is 8.46 s, while this is 20.4 s for the MILP solver.

6.5. Sensitivity analysis

To obtain insight into the sensitivity of the fitness function (13) to the weights, we conducted an experiment where we varied the weight of one specific component while fixing the value of the weights of the other components. That is, for a given instance, we selected the weight of one of the components (i.e., either travel time w_x, shift overtime w_y or waiting time w_z) and set it equal to the values in $\{1/10, 1/4, 1/3, 1/2, 1, 2, 3, 4, 10\}$, while keeping the other weights equal to 1. For each value in the set, we ran paGOMEA $R = 10$ times and computed the average of the actual (i.e., unweighted) value of the score components of the fitness function. This experiment is done for each of the three components. The experiment was performed on an instance created similarly as in Section 6.4, with the only difference being that the region is chosen at random and the number of activities on the instance is set to 24. The results of the analysis are illustrated in Fig. 9. From this it is clear that especially the waiting time is sensitive to changes in the weights, irrespective of the selected component weight which is varied. In particular, it seems to be the case that in order to make progress with respect to travel time or shift overtime, one has to make amends regarding the waiting time. However, the other way around, improving the waiting time does not have an equally strong impact on the travel time and shift overtime. Moreover, Fig. 9 also shows how the travel time and shift overtime relate to each other. That is, the travel time and waiting time either both increase together or decrease together, whereas they do not move in opposite directions. However, this is reasonable to expect as traveling between clients accounts for a large share of the total shift time.

7. Conclusions and further research

A novel model-based EA (paGOMEA) is presented that optimizes the daily planning of activities over a set of shifts for the HHCRSP. The goal of the optimization is to minimize a weighted sum of the travel time, time window waiting time and shift overtime, while respecting practical constraints such as qualification levels of the shifts. Next to the focus on optimization at the operational level, the tactical level is partly taken into account as well. In particular, the shift start times are selected in conjunction with the planned arrival times at the activities.

Numerical experiments based on real-life data showed that our paGOMEA leads to an objective value improvement of 41% compared to the current practice in a Dutch home health care organization. This improvement is established by intelligently guiding the optimization using a linkage tree model that is learned online. Our paGOMEA outperforms a more traditional EA that does not learn and exploit a model to guide the optimization. This demonstrates the potential of using machine learning to enhance the optimization of HHCRSP and permutation assignment problems in general.

Regarding the significant improvement compared to current practice, it should be taken into account that human planners presumably consider other components as well, e.g., continuity of care, a balanced workload, etc. A further line of research could thus be to quantify and include such components in the objective function to smoothen the adaptation in practice. It is also noteworthy that paGOMEA is able to generate a schedule in a matter of minutes, while this often takes hours for human planners.

Our proposed algorithm is flexible in the sense that it works for a larger range of fitness functions. A straightforward continuation of research is therefore to incorporate further components to the algorithm that consider the tactical level, e.g., from a pool of shifts choose an optimal selection that suits the planning of activities best.

The numerical comparisons between our paGOMEA and the minor-adapted pGOMEA counterpart showed the importance of understanding the types of dependencies that can occur and capturing these correctly in the dependency measure. Instead of manually constructing a suitable dependency measure for the problem at hand, an interesting approach to create more flexibility would be to apply learning techniques to automatically establish a dependency measure that is effective for optimizing the problem under consideration.

CRediT authorship contribution statement

Yoram Clapper: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualisation. **Joost Berkhout:** Conceptualization, Methodology, Software, Writing – review & editing, Visualisation. **René Bekker:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Dennis Moeke:** Conceptualization, Writing – original draft, Writing – review & editing, Funding acquisition.

Data availability

The authors do not have permission to share data.

Acknowledgment

We would like to thank Wouter Berkelmans for giving the inspiration to use GOMEA for permutation assignment problems and for giving us a first version of GOMEA code.

Funding

This research was funded by the Netherlands Organization for Scientific Research (NWO) under the Living Lab Sustainable Supply Chain Management in Healthcare project (project number: 439.18.457).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cor.2022.106081.

References

Aalvanger, G.H., Luong, N.H., Bosman, P.A.N., Thierens, D., 2018. Heuristics in permutation GOMEA for solving the permutation flowshop scheduling problem. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (Eds.), Parallel Problem Solving from Nature – PPSN XV. Springer International Publishing, Cham, pp. 146–157.

ActiZ, 2021. Infographic arbeidsmarkt VVT. URL https://www.actiz.nl/sites/default/files/2021-02/_DEF_ActiZ_Infographic%20Ontwikkeling%20Abeidsmarkt-VVT-versie2.0.pdf.

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.

Alp, G., Alkaya, A.F., 2022. An investigation of nature inspired algorithms on a particular vehicle routing problem in the presence of shift assignment. Comput. Oper. Res. 141, 105685.

Bean, J.C., 1994. Genetic algorithms and random keys for sequencing and optimization. ORSA J. Comput. 6 (2), 107–220.

Begur, S.V., Miller, D.M., Weaver, J.R., 1997. An integrated spatial DSS for scheduling and routing home-health-care nurses. Interfaces 27 (4), 35–48.

Bekker, R., Moeke, D., Schmidt, B., 2019. Keeping pace with the ebbs and flows in daily nursing home operations. Health Care Manag. Sci. 22 (2), 350–363.

Bennett, A.R., Erera, A.L., 2011. Dynamic periodic fixed appointment scheduling for home health. IIE Trans. Healthc. Syst. Eng. 1 (1), 6–19.

Benzarti, E., Sahin, E., Dallery, Y., 2013. Operations management applied to home care services: Analysis of the districting problem. Decis. Support Syst. 55 (2), 587–598.

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.

Bertsimas, D., Tsitsiklis, J.N., 1997. Introduction To Linear Optimization, Vol. 6. Athena Scientific Belmont, MA.

Bosman, P.A.N., Luong, N.H., Thierens, D., 2016. Expanding from discrete Cartesian to permutation gene-pool optimal mixing evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016. pp. 637–644.

Bosman, P.A.N., Thierens, D., 2013. More concise and robust linkage learning by filtering and combining linkage hierarchies. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation. pp. 359–366.

Bredström, D., Rönnqvist, M., 2008. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. European J. Oper. Res. 191 (1), 19–31.

Cheng, R., He, C., Jin, Y., Yao, X., 2018. Model-based evolutionary algorithms: a short survey. Complex Intell. Syst. 4, 283–292.

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. Healthc. 13, 1–22.

Cover, T.M., 1999. Elements of Information Theory. John Wiley & Sons.

de Oliveira da Costa, P.R., Mauceri, S., Carroll, P., Pallonetto, F., 2018a. A genetic algorithm for a green vehicle routing problem. Electron. Notes Discrete Math. 64, 65–74.

de Oliveira da Costa, P.R., Mauceri, S., Carroll, P., Pallonetto, F., 2018b. A genetic algorithm for a green vehicle routing problem. Electron. Notes Discrete Math. 64, 65–74. http://dx.doi.org/10.1016/j.endm.2018.01.008, URL https://www.sciencedirect.com/science/article/pii/S1571065318300088. 8th International Network Optimization Conference - INOC 2017.

Decerle, J., Grunder, O., El Hassani, A.H., Barakat, O., 2018. A memetic algorithm for a home health care routing and scheduling problem. Oper. Res. Health Care 16, 59–71.

Decerle, J., Grunder, O., El Hassani, A.H., 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.

Di Mascolo, M., Martinez, C., Espinouse, M.-L., 2021. Routing and scheduling in home health care: A literature survey and bibliometric analysis. Comput. Ind. Eng. 158, 107255.

Dieleman, N.A., Buitink, M., Bekker, R., Moeke, D., 2022. A three-step framework for capacity planning in a nursing home context. Health Syst. 1–18.

Eiben, A.E., Smith, J.E., 2015. Introduction To Evolutionary Computing, second ed. Springer.

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

Grenouilleau, F., Lahrichi, N., Rousseau, L.-M., 2020. New decomposition methods for home care scheduling with predefined visits. Comput. Oper. Res. 115, 104855.

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

Grieco, L., Utley, M., Crowe, S., 2021. Operational research applied to decisions in home health care: A systematic literature review. J. Oper. Res. Soc. 72 (9), 1960–1991.

Gu, H., Zhang, Y., Zinder, Y., 2022. An efficient optimisation procedure for the Workforce Scheduling and Routing Problem: Lagrangian relaxation and iterated local search. Comput. Oper. Res. 144, 105829.

Guermazi, R., Chaabane, I., Hammami, M., 2018. AECID: Asymmetric entropy for classifying imbalanced data. Inform. Sci. 467, 373–397.

Hans, E.W., Houdenhoven, M.v., Hulshof, P.J.H., 2012. A framework for healthcare planning and control. In: Handbook of Healthcare System Scheduling. Springer, pp. 303–320.

Hewitt, M., Nowak, M., Nataraj, N., 2016. Planning strategies for home health care delivery. Asia-Pac. J. Oper. Res. 33 (05), 1650041.

Karakatič, S., Podgorelec, V., 2015. A survey of genetic algorithms for solving multi depot vehicle routing problem. Appl. Soft Comput. 27, 519–532.

Knight, S., Tjassing, H., 1994. Health care moves to the home. World Health 4, 413–444.

Lanzarone, E., Matta, A., 2014. Robust nurse-to-patient assignment in home care services to minimize overtimes under continuity of care. Oper. Res. Health Care 3 (2), 48–58.

Lin, C.-C., Hung, L.-P., Liu, W.-Y., Tsai, M.-C., 2018. Jointly rostering, routing, and rerostering for home health care services: A harmony search approach with genetic, saturation, inheritance, and immigrant schemes. Comput. Ind. Eng. 115, 151–166.

Liu, M., Yang, D., Su, Q., Xu, L., 2018. Bi-objective approaches for home healthcare medical team planning and scheduling problem. Comput. Appl. Math. 37 (4), 4443–4474.

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.

Matta, A., Chahed, S., Sahin, E., Dallery, Y., 2014. Modelling home care organisations from an operations management perspective. Flex. Serv. Manuf. J. 26 (3), 295–319.

Mısır, M., Smet, P., Vanden Berghe, G., 2015. An analysis of generalised heuristics for vehicle routing and personnel rostering problems. J. Oper. Res. Soc. 66 (5), 858–870.

Müllner, D., 2011. Modern hierarchical, agglomerative clustering algorithms. http://dx.doi.org/10.48550/ARXIV.1109.2378, URL https://arxiv.org/abs/1109.2378.

Nasir, J.A., Dang, C., 2018. Solving a more flexible home health care scheduling and routing problem with joint patient and nursing staff selection. Sustainability 10 (1), 148.

Nguyen, T.V.L., Toklu, N.E., Montemanni, R., 2015. Matheuristic optimization for robust home health care services. In: Proc. ICAOR. p. 2.

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.

NIDI/CBS, 2020. Bevolking 2050 in beeld Drukker, diverser en dubbelgrijs: Deelrapport Verkenning Bevolking 2050. URL https://publ.nidi.nl/output/2020/nidi-cbs-2020-bevolking-2050-in-beeld.pdf.

Oladzad-Abbasabady, N., Tavakkoli-Moghaddam, R., 2022. Dynamic routing-scheduling problem for home health care considering caregiver-patient compatibility. Comput. Oper. Res. 106000.

Przewozniczek, M.W., Komarnicki, M.M., Bosman, P.A.N., Thierens, D., Frej, B., Luong, N.H., 2021. Hybrid linkage learning for permutation optimization with gene-pool optimal mixing evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. GECCO '21, Association for Computing Machinery, New York, NY, USA, pp. 1442–1450. http://dx.doi.org/10.1145/3449726.3463152.

Rasmussen, M.S., Justesen, T., Dohn, A., Larsen, J., 2012. The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies. European J. Oper. Res. 219 (3), 598–610.

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.

Restrepo, M.I., Rousseau, L., Vallée, J., 2020. Home healthcare integrated staffing and scheduling. Omega 95, 102057.

Rodriguez, C., Garaix, T., Xie, X., Augusto, V., 2015. Staff dimensioning in homecare services with uncertain demands. Int. J. Prod. Res. 53 (24), 7396–7410.

Rodriguez-Verjan, C., Augusto, V., Xie, X., 2018. Home health-care network design: Location and configuration of home health-care centers. Oper. Res. Health Care 17, 28–41.

Schrotenboer, A.H., uit het Broek, M.A.J., Jargalsaikhan, B., Roodbergen, K., 2018. Coordinating technician allocation and maintenance routing for offshore wind farms. Comput. Oper. Res. 98, 185–197.

SER, 2020. Zorg voor de toekomst: Over de toekomstbestendigheid van de zorg. URL https://www.fbz.nl/wp-content/uploads/2020/06/SER-rapport-zorg-voor-de-toekomst-juni-2020.pdf.

Thierens, D., 2010. The linkage tree genetic algorithm. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (Eds.), Parallel Problem Solving from Nature, PPSN XI. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 264–273.

Thierens, D., Bosman, P.A.N., 2012. Predetermined versus learned linkage models. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation. GECCO '12, Association for Computing Machinery, New York, NY, USA, pp. 289–296. http://dx.doi.org/10.1145/2330163.2330205.

Trautsamwieser, A., Hirsch, P., 2014. A Branch-Price-and-Cut approach for solving the medium-term home health care planning problem. Networks 64 (3), 143–159.

Yadav, N., Tanksale, A., 2022. An integrated routing and scheduling problem for home healthcare delivery with limited person-to-person contact. European J. Oper. Res..

Yuan, B., Liu, R., Jiang, Z., 2015. A branch-and-price algorithm for the home health care scheduling and routing problem with stochastic service times and skill requirements. Int. J. Prod. Res. 53 (24), 7450–7464.

Zhan, Y., Wan, G., 2018. Vehicle routing and appointment scheduling with team assignment for home services. Comput. Oper. Res. 100, 1–11.