Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Elsevier hereby grants permission to make all its COVID-19-related research that is available on the COVID-19 resource centre - including this research content - immediately available in PubMed Central and other publicly funded repositories, such as the WHO COVID database with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the COVID-19 resource centre remains active.
Staff scheduling for residential care under pandemic conditions: The case of COVID-19

Amirhossein Moosavi\textsuperscript{a,}*, Onur Ozturk\textsuperscript{a}, Jonathan Patrick\textsuperscript{a}

University of Ottawa, Telfer School of Management, 55 Laurier Avenue East, Ottawa, Ontario K1N 6N5, Canada

A R T I C L E   I N F O

Article history:
Received 7 September 2021
Accepted 30 April 2022
Available online 24 May 2022

Keywords:
Residential care
Staff scheduling
Pandemic condition
COVID-19
Advanced analytics
Heuristic
Multi-objective optimization

A B S T R A C T

The COVID-19 pandemic severely impacted residential care delivery all around the world. This study investigates the current scheduling methods in residential care facilities in order to enhance them for pandemic conditions. We first define the basic problem that addresses decisions associated with the assignment and scheduling of staff members, who perform a set of tasks required by residents during a planning horizon. This problem includes the minimization of costs associated with the salary of part-time staff members, total overtime, and violations of service time windows. Subsequently, we adapt the basic problem to pandemic conditions by considering the impacts of communal spaces (e.g., shared rooms) and a cohorting policy (classification of residents based on their risk of infection) on the spread of infectious diseases. We introduce a new objective function that minimizes the number of distinct staff members serving each room of residents. Likewise, we propose a new objective function for the cohorting policy that aims to minimize the number of distinct cohorts served by each staff member. A new constraint is incorporated that forces staff members to serve only one cohort within a shift. We present a population-based heuristic algorithm to solve this problem. Through a comparison with two benchmark solution approaches (a mathematical programming and a non-dominated arch colony optimization algorithm), the superiority of the heuristic algorithm is shown regarding solution quality and CPU time. Finally, we conduct numerical analyses to present managerial implications.

© 2022 Elsevier Ltd. All rights reserved.

1. Introduction

Residential Care (RC), sometimes called as institutional/long-term care, assists adults and children with various traditional health services (e.g., mental health, complex chronic, disability, Alzheimer, dementia and hospice) and assistive care services (e.g., caregiving and social support) [68]. RC services are usually delivered in the form of (a) facility-based care, (b) community-based care, and (c) home-based care [15]. Facility-based care refers to facilities/homes that provide 24-hour nursing and personal care for their residents. Community-based care provides people in need with communal and pre-planned services, while home-based care enables them to receive care services in their homes. Given that such services are mainly used by seniors, world population aging may cause enormous challenges in providing on-time RC, where RC Facilities (RCFs) play an important role.

In Canada, the demand for RCFs exceeds the current capacity resulting in wait times for new admissions [34]. Two reasons hinder the increase of capacity in the RC sector. First, according to Statistics Canada [65], the building and operation of RCFs are expensive. They estimated that new RC beds will cost $64 billion to construct and $130 billion to operate by 2035 (cumulative costs). Second, adequate staffing of RCFs is a major challenge [10]. Therefore, we need to use available Staff Members (SMs) efficiently in order to assure the quality and safety of care as well as to ensure that we can serve as many patients as possible. Note that this problem is usually termed as the staff scheduling problem.

Moreover, the COVID-19 pandemic has increased the complexity of service delivery in RCFs. The World Health Organization declared COVID-19 a pandemic in February of 2020 [50]. As of the 20\textsuperscript{th} of April 2022, the global confirmed cases of COVID-19 surged to more than 500 million patients [69]. The spread of this virus has significantly impacted the delivery of service in RCFs in Canada [17]. Estimates suggest that 62–82% of deaths in Canada due to
COVID-19 have occurred among residents of RCFs [17,48], a reality that has received abundant media coverage and generated a lot of policy discussion [3]. Unfortunately, what has occurred in RCFs is a national catastrophe. Many deaths may have been prevented if enough attention was paid to preparing the RC sector [39]. The lack of efficient planning, including staff scheduling, has revealed significant inadequacies within this sector [7,29]. Like Canada, RCFs in several other developed countries, such as the United States, the United Kingdom, Spain, etc., have suffered from the COVID-19 pandemic [63]. Therefore, this study aims to adapt the current scheduling methods used in RCFs for pandemic conditions, aiming to mitigate the impact of COVID-19 on these facilities, while ensuring the optimum allocation of resources.

The remainder of this paper is organized as follows. We first review the literature in Section 2. Then, we present a problem description of staff scheduling for residential care during a pandemic and formulate it as a Mixed-Integer Linear Programming (MILP) model in Section 3. Since the problem we study is NP-hard, in Section 4, we propose a fast and efficient population-based heuristic algorithm that constructs initial solutions with two stages - assignment and scheduling of SMs to tasks - and applies a repairing mechanism and a local search method to ensure the feasibility and high-quality of solutions, respectively. Section 5 presents our experimental results, which evaluates (a) the efficiency of the proposed heuristic algorithm in comparison to two benchmark solutions approaches, and (b) different features of the problem under consideration used to extract managerial implications. Finally, Section 6 provides conclusions and directions for future research.

2. Literature review

In this section, we first review some variants of staff scheduling problems, with an emphasis on the RC settings. Then, we review staff scheduling problems under pandemic conditions to compare with our study.

2.1. Staff scheduling

Staff scheduling problems have been thoroughly studied during the previous few decades. Economic concerns could be driving the increase in research attention to such problems. Labor costs are a substantial direct cost component for many businesses. Implementing a new staff schedule to cut these costs by just a few percent might be quite advantageous [8]. The staff scheduling problem that Dantzig [19] introduced in the 1950s is different from today’s problems. The importance of meeting employee requirements has significantly risen in staffing and scheduling decisions. When establishing work schedules, companies offer part-time contracts or flexible work hours and take into consideration employee preferences (e.g., working with others, preferring a specific shift type, and specified days off or on).

Guo and Bard [37] addressed a bi-objective staff scheduling problem using an augmented MILP-based methodology (a hybrid MILP formulation and column generation algorithm). The authors deployed the weighted sum method to minimize the uncovered demand and staff preference violations. Aydás et al. [5] evaluated a short-term adjustment staff scheduling problem under demand uncertainties. To meet the demand, Aydás et al. modeled six different adjustment options for a two-stage stochastic programming model and optimized the total costs of staffing and adjustments to the original schedules. Smet et al. [64] studied a staff scheduling problem where tasks must be assigned to a set of heterogeneous SMs during a multi-shift planning horizon. They considered non-preemptive fixed tasks and shifts. The authors proposed three constructive heuristics and a neighborhood search algorithm in order to minimize the total scheduling costs. Likewise, Hojati [38] investigated a staff scheduling problem in which heterogeneous SMs are assigned to a group of known tasks and shifts within a planning horizon. The authors developed an iterative greedy algorithm and demonstrated that the greedy algorithm can find high-quality solutions. Maenhout and Vannoucke [49] integrated the staffing and scheduling decisions in a health center while considering nurse characteristics. They formulated this problem based on the Dantzig-Wolfe decomposition approach to optimize the effectiveness in providing care, efficiency of a nursing unit and job satisfaction among staff.

Staff scheduling problems in RC settings have been primarily dedicated to home- and facility-based care (the community-based care is mostly ignored). Such problems for home- and facility-based care are similar with a few main differences. In home-based care, residents receive care services in their homes and SMs usually incur considerable travel times for serving residents [13]. There are three types of rooms available for residents in facility-based care (private, semi-private, and shared), and SMs’ travel times are generally insignificant [34]. Due to their similarity, we review both of them in the following.

Home-based care:

Demirbilek et al. [24] studied a dynamic and multi-staff routing and scheduling problem that maximizes the number of visits during a planning horizon. Given present and unknown future requests, the problem decide (a) whether to accept the future requests and (b) how to assign and schedule SMs to accepted requests. The authors developed a population-based heuristic algorithm, which significantly outperforms an existing greedy algorithm. Cinar et al. [16] investigated a multi-period home-based care routing and scheduling problem for a single SM. Assigning priorities to patients based on their last visit time and the severity of their condition, the problem’s objective is to minimize the total priority of visited patients. To find near-optimal solutions for large-sized instances, the authors proposed an adaptive large neighborhood search algorithm and a metaheuristic. According to numerical analysis, the latter algorithm could provide higher quality solutions, while the former one outperformed in terms of CPU time.

Méndez-Fernández et al. [52] addressed a staff routing and scheduling problem in a home-based care system over a planning horizon. In this problem, the authors considered the preference of patients, task time windows and travel times between every two patients, while maximized the continuity of care for patients. They proposed a simulated annealing algorithm and demonstrated its efficiency in solving real-life instances. Frifita and Masmoudi [30] investigated a home-based care routing and scheduling problem with time window, heterogeneous SMs, and temporal dependencies (synchronization, precedence, and disjunction constraints). The main goal of this research is to serve a set of geographically diverse patients such that visiting costs are minimized. Frifitaa and Masmoudi presented three variants of the variable neighborhood search algorithm and showed their efficiency in comparison with benchmark algorithms. Restrepo et al. [60] addressed the staffing and scheduling problem in home healthcare using a two stage stochastic programming model to evaluate demand uncertainties. Through a comparison with the deterministic version of the problem, Restrepo et al. demonstrated that the stochastic model leads to considerable cost savings since its decisions are more robust to accommodate changes in demand. Mosquera et al. [55] studied a home care scheduling problem with flexible task durations. The authors proposed a two-phase methodology. A greedy heuristic first generates a solution, which is then enhanced by a local search algorithm exploring various neighborhoods.

Facility-based care:

Facility-based care scheduling problems usually include decisions associated with the assignment (instead of routing for home-based care settings) and scheduling of SMs working in an RCF. Fo-
cusing on the staff scheduling problem, Koeleman et al. [42] modeled an RCF system as a Markov decision process in order to evaluate its monotonicity properties. Based on the extracted insights, the authors developed a trunk reservation heuristic to schedule SMs in an RCF and demonstrated its near-optimal performance, even for large-sized instances. Lieder et al. [44] studied a task-based scheduling problem in RCFs. They proposed an MILP model and a dynamic program to minimize the total tardiness and earliness penalties of all tasks considering two main task categories - known and unknown tasks. Bagheri et al. [6] presented a stochastic programming model to investigate a staff scheduling problem that accounts for demand uncertainties and the lengths of stay of patients. They applied the sample average approximation method to solve the stochastic model and find optimal solutions.

2.2. Scheduling under pandemic conditions

The COVID-19 virus is highly contagious, and its spread has caused considerable financial, ecological, and social difficulties. Under this circumstance, it is extensively important to protect healthcare professionals and those who are vulnerable to the COVID-19 virus (chiefly seniors). In addition to the provision of appropriate personal protective equipment, healthcare service delivery should be planned cautiously. In the following, we present a review of scheduling problems under pandemic conditions.

On-time healthcare delivery has faced difficulties under pandemic conditions. Fujita et al. [31] assessed the COVID-19 pandemic’s impact on lung cancer treatment scheduling. They retrospectively evaluated the medical records of patients at a national hospital in Kyoto, Japan. Fujita et al. found that 9.1% of patients experienced a delay in lung cancer treatment during the COVID-19 pandemic. Kluger et al. [40] used the Monte Carlo simulation to explore different staffing policies and minimize infections among SMs that work on non-COVID-19 wards. They reported that longer SM shifts and less co-rotation of SMs (no more frequently than every three days) may lead to fewer infections. Güler and Gebcic [36] used mathematical programming to investigate a days off scheduling problem for physicians under pandemic conditions. In an effort to decrease the spread of COVID-19 virus, their model incorporates a soft constraint to secure two consecutive off days for each physician after a working shift. Thus, their model minimizes the violations from the aforementioned constraint while considering the availability of physicians. Abadi et al. [1] also studied the days off scheduling problem for SMs that serve COVID-19 patients. They assumed that eliminating unbalanced workloads and overtime for SMs can improve their performance and decrease their errors, by which they can work safer under pandemic conditions. Abadi et al. proposed a hybrid salp swarm algorithm and genetic algorithm to solve large-sized instances of this problem. Their findings suggested that the proposed solution approach outperforms the state-of-the-art solution approaches.

In addition to the healthcare environment, scheduling problems under pandemic conditions are adapted for industrial and business environments as well. Zucchi et al. [73] studied a staff scheduling problem for an Italian pharmaceutical distribution warehouse in the context of COVID-19 pandemic. While considering the contractual working time of SMs, Zucchi et al. assumed that SMs must be divided into mutually exclusive groups in order to decrease the circulation of COVID-19 virus. They attained an optimal schedule for this problem using an MILP model, which was shown to be better than the one generated manually. Guerriero and Guido [35] presented six integer programming models to investigate the staff scheduling problem under pandemic conditions with various flexibility levels. The proposed models consider demand requirements, SMs-personal and family responsibilities, and anti COVID-19 measures (e.g., SMs can work both on-site and remotely). Using real data provided by different departments at the University of Calabria, Guerriero and Guido demonstrated the superiority of the schedules constructed by their models compared to the ones built manually.

In contrast to the literature, the main contributions of our investigation are three-fold. (a) Unlike previous papers, we study a staff scheduling problem under pandemic conditions that includes both days off scheduling and shift scheduling. (b) We take into account the effects of communal spaces and a cohorting policy (i.e., categorizing patients based on the infection risk, further details in Section 3) on the spread of respiratory prone diseases. Thus, we propose two new objective functions and a new constraint set that aim to mitigate the circulation of infectious viruses (e.g., COVID-19). (c) We propose an efficient and versatile heuristic algorithm that outperforms two well-known benchmark solution approaches over a wide variety of instances in terms of both solution quality and CPU time.

3. Staff scheduling for residential care

In this section, we aim to discuss and formulate the staff scheduling problem for RC under both non-pandemic and pandemic conditions. Sub-Section 3.1 describes a staff scheduling problem for RC under non-pandemic conditions and formulates it using an MILP model. Sub-Section 3.2 introduces new assumptions and objectives to adapt the problem for pandemic conditions. Finally, Sub-Section 3.3 provides an analysis of the complexity of the proposed staff scheduling problem.

3.1. Non-pandemic conditions

Given sets of predefined tasks \((i \in \mathcal{T})\) and SMs \((c \in \mathcal{C})\), the main decision is the allocation of the SMs to the tasks needed by the residents during a planning horizon of \(|\mathcal{D}|\) days \((d \in \mathcal{D})\) (the length of a planning horizon is assumed to be one week in this study). We assume that each day includes a number of shifts \((s \in \mathcal{S})\) with preset beginning and finishing times (one, two or three shifts per day). In addition to predefined tasks required on a daily or weekly basis (e.g., physical assessment, monitoring blood pressure, routine hygiene, and oral care), residents may need non-predefined tasks (e.g., mental health emergency/heart failure recovery, triage incoming patient calls, and toileting care). Therefore, we assume that the total working time of each SM used to serve predefined tasks on each shift must be limited to a percentage of the shift’s regular length (this limitation can be relaxed based on the qualification level of SMs). Allowing overtime with a penalty, SMs would be able to serve tasks even after the regular length of a shift. Given that we discretize time into time slots, overtime for each SM cannot exceed a preset number of time slots \((mO\text{ time slots})\). Overtime of an SM is equal to the number of time slots that she is working after the regular length of a shift. SMs are allowed to work one shift within each day, and the maximum number of shifts they can work during a planning horizon is limited to a predefined value \((mS\text{ shifts})\).

We assume that RCFs can use permanent SMs or hire part-time SMs (with an employment cost per shift) to serve residents. Both permanent and part-time SMs may have different availability preferences throughout a planning horizon. We assume that tasks have predetermined service times, and their start times are desired to be within preset time intervals. Start times outside these intervals - earliness and lateness - can occur with a penalty (just like due date violations in scheduling problems). The earliness and lateness of tasks are computed based on the number of time slots that their start times are removed from the associated intervals. Tasks are heterogeneous, meaning that they can only be served by SMs with the appropriate qualification level. Serving some of the tasks may
require the simultaneous involvement of more than one SM (synchronized tasks and SMs).

Finally, we consider one objective for this problem, which reduces the costs of hiring part-time SMs, overtime of all SMs and violation of tasks’ starting times from their time windows. To formulate an MILP model for the aforementioned problem, the required indices, sets, parameters, and decision variables are defined as follows:

| - Indices:                      |
|-------------------------------|
| $i$, $j$, $n$                  | Indices of tasks.                      |
| $c$, $h$                      | Indices of SMs.                        |
| $s$                           | Index of shifts for all days of the planning horizon. |
| $d$                           | Index of days.                         |

- Sets:

| $\alpha$                      | Singleton set used to determine the start/end of SMs work on each shift (a dummy task). |
|-------------------------------|
| $T$                           | Set of all tasks excluding the dummy task. |
| $C$                           | Set of all SMs. |
| $C'$                          | Set of part-time SMs ($C' \subseteq C$). |
| $S$                           | Set of all shifts. |
| $D$                           | Set of all days. |

- Parameters:

| $U_{sd}$                      | If shift $s$ belongs to day $d$, 1; otherwise, 0. |
|-------------------------------|
| $BS_s$, $ES_s$                | Beginning and end of shift $s$ (overtime excluded), respectively. |
| $mO$                          | Maximum overtime allowed for each SM per shift (measured based on time slots). |
| $mS$                          | Maximum number of shifts each SM is allowed to work during a planning horizon. |
| $fR_c$                        | The maximum percentage of a shift’s regular length that SM $c$ is allowed to work serving predefined tasks. |
| $p_{nc}$                      | Salary of part-time SM $c$ per shift. |
| $p_{nc}$                      | Overtime cost of SM $c$ per time slot. |
| $p_{nc}$                      | Penalty of task $i$ on shift $s$ for each time slot of violation from its time window. |
| $AP_c$                        | If SM $c$ is available to serve residents on shift $s$, 1; otherwise, 0. |
| $Q_{ic}$                      | Qualification level required to serve task $i$ on shift $s$. |
| $Q_{ic}$                      | Qualification level of SM $c$. |
| $ST_s$                        | Service time of task $i$ on shift $s$ (measured based on time slots). |
| $NC_s$                        | Number of SMs required by task $i$ on shift $s$. |
| $ET_{is}$, $LT_{is}$          | Earliest and latest preferred times for starting task $i$ on shift $s$, respectively. |
| $M$                           | An adequately large number. |

- Decision Variables:

| $x_{nc}$                      | If SM $c$ serves task $j$ right after task $i$ on shift $s$, 1; otherwise, 0. |
|-------------------------------|
| $a_{nc}$                      | Start time of serving task $i$ by SM $c$ on shift $s$. |
| $e_{nc}$, $l_{nc}$            | The violations from the earliest and latest preferred times of starting task $i$ on shift $s$ (measured based on time slots), respectively. |
| $o_{nc}$                      | Overtime occurred for SM $c$ on shift $s$ (measured based on time slots). |
The objective function is formulated as follows:

\[ \text{Min } F_1 = \sum_{j \in T} \sum_{c \in C} \sum_{s \in S} P_{jc} x_{a_j cs} + \sum_{j \in T} \sum_{c \in C} \sum_{s \in S} P_{jc} o_{cs} + \sum_{j \in T} \sum_{c \in C} \sum_{s \in S} P_{jbs} (e_{bs} + I_{bs}) \]  

(1)

Objective function (1) minimizes the total costs of hiring part-time SMs, SMs’ overtime and penalties for violations from the time windows of all tasks (both the earliest and latest preferred starting times). This objective function is subjected to the following constraints.

\[ \sum_{i \in T} \sum_{c \in C} x_{ijcs} = NC_{jcs} \quad \forall j \in T; c \in C; s \in S \]  

(2)

\[ \sum_{i \in T} (Q_{jcs} - Q_{c}) x_{ijcs} = 0 \quad \forall j \in T; c \in C; s \in S \]  

(3)

\[ \sum_{i \in T} x_{ijcs} \leq AP_{cs} \quad \forall j \in T; c \in C; s \in S \]  

(4)

Constraint set (2) makes sure that all tasks are served throughout the planning horizon. Serving a task may require one or more SMs. Thus, if task \( j \) on shift \( s \) requires \( NC_{jcs} \) SMs, this constraint set guarantees that enough SMs are assigned to the task. Constraint set (3) ensures that SMs allocated to task \( j \) on shift \( s \) have the appropriate qualification level. Constraint set (4) also assures that SMs are allocated to shifts for which they are available to work.

\[ \sum_{i \in T} \sum_{c \in C} U_{ics} x_{a_j cs} \leq 1 \quad \forall c \in C; d \in D \]  

(5)

\[ \sum_{i \in T} \sum_{c \in C} x_{a_j cs} \leq mS \quad \forall c \in C \]  

(6)

Constraint set (5) guarantees that each SM can work at most one shift within a day. Constraint set (6) ensures that the total number of shifts each SM can work within a planning horizon must be smaller than or equal to \( mS \).

\[ \sum_{i \in T} x_{ijcs} \leq \sum_{i \in T} x_{a_jcs} \quad \forall j \in T; c \in C; s \in S \]  

(7)

\[ \sum_{i \in T} x_{ijcs} \leq \sum_{i \in T} x_{a_jcs} \quad \forall j \in T; c \in C; s \in S \]  

(8)

\[ \sum_{i \in T} x_{ijcs} - \sum_{i \in T} x_{ijcs} = 0 \quad \forall j \in T; c \in C; s \in S \]  

(9)

Constraint sets (7) and (8) force SMs to start and finish working on each shift by doing the dummy task. Constraint set (9) guarantees that after serving a task, the SMs must start doing another task or finish working by doing the dummy task.

\[ x_{ijcs} = 0 \quad \forall i, j \in T \cup \alpha; c \in C; s \in S; i = j \]  

(10)

\[ \sum_{i \in T} x_{ijcs} \leq 1 \quad \forall i \in T \cup \alpha; c \in C; s \in S \]  

(11)

Constraint set (10) guarantees that SMs cannot do a specific task twice consecutively, which prevents \( x_{a_jcs} = 1 \). Constraint set (11) ensures that SMs can only start serving one task right after finishing another task and a specific task cannot be performed more than once by the same SM.

\[ a_{a_jcs} = a_{jcs} + ST_{jcs} - M (1 - x_{ijcs}) \quad \forall i \in T \cup \alpha; c \in C; s \in S \]  

(12)

\[ \sum_{i \in T} ET_{jcs} x_{ijcs} - e_{jcs} \leq a_{jcs} \quad \forall j \in T; c \in C; s \in S \]  

(13)

\[ a_{jcs} \geq a_{jcs} + ST_{jcs} - M (1 - x_{ijcs}) \quad \forall j \in T; c \in C; s \in S \]  

(14)

If task \( j \) is served right after task \( i \) by SM \( c \) on shift \( s \). Constraint set (12) assures that the start time of serving task \( j \) must be greater than the start time of serving task \( i \) plus its service time. Constraint sets (13) and (14) guarantee that serving task \( j \) should start within its time window (a time interval between \( ET_{jcs} \) and \( LT_{jcs} \)). Otherwise, a violation from the time window occurs \( (e_{jcs} \) for earliness and \( I_{jcs} \) for lateness).

\[ a_{jcs} \geq BS_{jcs} - M \left( \sum_{j \in T \cup \alpha} x_{jcs} \right) \quad \forall i \in T; c \in C; s \in S \]  

(15)

\[ a_{jcs} + ST_{jcs} - M (1 - x_{a_jcs}) \leq ES_{jcs} + o_{cs} \quad \forall i \in T; c \in C; s \in S \]  

(16)

\[ o_{cs} \leq mO \quad \forall c \in C; s \in S \]  

(17)

\[ a_{jcs} - a_{jcs} \geq M ( \sum_{i \in T \cup \alpha} x_{ijcs} + \sum_{i \in T \cup \alpha} x_{ijcs} - 2) \quad \forall j \in T; b, c \in C; s \in S; NC_{jcs} > 1 \]  

(18)

\[ \sum_{i \in T \cup \alpha} \sum_{j \in T} ST_{jcs} x_{ijcs} \leq fR_{c} (ES_{jcs} - BS_{jcs}) \quad \forall c \in C; s \in S \]  

(19)

Constraint set (15) ensures that all tasks on shift \( s \) cannot be served sooner than the beginning of the shift. Constraint set (16) guarantees that the end time of serving all tasks on shift \( s \) must be sooner than the end of the shift unless overtime occurs. Constraint set (17) ensures that the overtime of each SM on shift \( s \) must be less than or equal to \( mO \). Constraint set (18) assures that synchronized SMs assigned to task \( j \) on shift \( s \) must start serving the task at the same time. Constraint set (19) makes sure that the total duration of tasks allocated to an SM on a shift must be less than or equal to \( fR_{c} \) of shift’s regular length. It is worth mentioning that a very large or small value for \( M \) in Constraint sets (12), (15), (16) and (18) might cause rounding error or infeasibility, respectively. Therefore, it is critical to choose an appropriate value for \( M \). In Constraint sets (12), (16) and (18), an appropriate value for \( M \) is \( \max_{s \in \mathbb{S}} (ES_{jcs} + mO + 1) \). On the other hand, an appropriate value for \( M \) in Constraint set (15) is \( \max_{s \in \mathbb{S}} (BS_{jcs} + 1) \). Therefore, \( M \) should be set equal to \( \max_{s \in \mathbb{S}} (ES_{jcs} + mO + 1) \), \( \max_{s \in \mathbb{S}} (BS_{jcs} + 1) \) in all constraints.

\[ x_{ijcs} \in \{0, 1\} \quad \forall j \in T \cup \alpha; c \in C; s \in S \]  

(20)

Constraint set (20) defines the binary variables. And Constraint set (21) specifies the non-negativity of the continuous variables.

3.2. Pandemic conditions

The transmission of COVID-19 mainly occurs due to person-to-person contact through (a) droplets/aerosol and (b) contact with a contaminated surface [14,20]. With this in mind, there are six potential ways for the transmission of this virus among residents and SMs, (a) a resident is infected by SMs, (b) a resident is infected by other residents, (c) an SM is infected by residents, (d) an SM is infected by other SMs, (e) an SM is infected outside of RCFs and (f) residents are infected by visitors. Given the vulnerability of residents in RCFs during pandemic conditions [27,32], infection prevention strategies are considered as the most effective approaches.
to reduce overall fatality in this population [45]. Therefore, in the following, we introduce new assumptions for the staff scheduling problem in RC settings and propose two new objectives to control the transmission of COVID-19 (or any other transmissible infection) among residents and SMs.

In RCFs, there are usually three types of rooms for residents, (a) basic-which is occupied by two (or more) residents and has a shared washroom, (b) semi-private-which accommodates one resident and has a washroom shared by another resident in a nearby semi-private room, and (c) private-which is occupied by one resident and has a private washroom. The first two types of rooms have communal space which has long been known as a key reason for the susceptibility of RCFs to pandemics [32]. To control the spread of COVID-19, Liu et al. [45] suggested the transition of existing facilities away from shared rooms (e.g., using basic rooms as private ones). This policy could be effective in preventing COVID-19 infections in RCFs. However, it would result in the sub-optimal use of available resources, particularly space and budget, which is undesirable/impossible for RCFs that suffer from a shortage of space [70]. We design an objective function to reduce the number of distinct SMs serving each room. Not only does the application of this objective decrease the risk of spread of COVID-19 among residents, but it also increases the continuity of care by decreasing the variability of SMs serving each room which is desirable even in non-pandemic conditions. It is worth noting that we consider semi-private rooms as one room since they share a communal space. These rooms should be ideally served by the same SMs.

To prevent the spread of COVID-19 in RCFs, Public Health Ontario issued an outbreak guideline in April 2020 to encourage RCFs to group residents according to whether they have tested positive for COVID-19 or their risk of infection during an outbreak [59]. Based on this guideline, residents are grouped into four cohorts. (a) not exposed and well cohort, (b) exposed and well cohort, (c) exposed, ill, but not known to have COVID-19 cohort, and (d) COVID-19 positive and infectious cohort. Note that exposed here refers to residents who have been in touch with infected SMs/residents. In addition to cohorting residents, this guideline encourages RCF management to assign SMs to only work with one cohort within a shift in order to decrease their movement between different cohorts [59]. According to Brown et al. [9], Li et al. [43], Rios et al. [61], application of this guideline has proven to be an effective measure in RCFs to prevent the spread of COVID-19. Accordingly, we first assume that residents are grouped into multiple cohorts and do not move between different cohorts during the planning horizon. Also, each SM can visit only one cohort of residents within a shift. However, if a resident moves between cohorts during a planning horizon, one policy is to eliminate tasks related to them from the incumbent schedule and manually schedule the tasks (more suitable when a few residents change their cohorts). Another policy is to use our proposed approach again (at the end of the disruption day) and find an up-to-date schedule based on new inputs for the rest of the planning horizon (more suitable when several residents change their cohorts). If SMs get sick during the planning horizon, then again, our approach should be relaunched with the updated data to find a new schedule. If no feasible solution is found with the remaining SMs, then, the RC management should negotiate with current SMs to increase their availability, take the decision to hire additional SMs or re-evaluate tasks.

Secondly, we consider the preference of SMs in working with different cohorts (e.g., not serving the COVID-19 positive cohort). One approach for considering the preference of SMs is to guarantee that their preferences are strictly respected (i.e., SMs will be never assigned to the COVID-19 positive cohort if they prefer not serving them). Another approach is to allow the violation of SMs’ preference while minimizing the violations. Although the latter approach is more flexible, it is more complex to implement because: (a) it adds one objective function to the optimization problem, and (b) it requires a weighting method in order to measure the violation of the SMs’ preference for different cohorts (with different weights). The former approach is easier to implement and more effective to prevent the absenteeism of SMs due to fear of COVID-19 [29]. In this study, we deploy the former approach, however, practitioners can adapt the latter one depending on their preference. Lastly, in line with the above guideline, another objective function could be the minimization of the number of distinct cohorts visited by an SM within a planning horizon. This objective reduces the movement of SMs among different cohorts, aiming to decrease the possibility of COVID-19 spread among both SMs and cohorts.

To incorporate the above assumptions and objectives into the non-pandemic model, we first define new notations as follows.

- **Indices:**
  - \( r, k \) Indices of rooms.
  - \( h, m \) Indices of cohorts.

- **Sets:**
  - \( R \) Set of all rooms.
  - \( H \) Set of all cohorts.

- **Parameters:**
  - \( A_{Ph} \) If SM \( c \) is available to serve residents of cohort \( h \) on shift \( s \), \( 1 \); otherwise, 0.
  - \( V_{kr} \) If task \( i \) on shift \( s \) belongs to a resident who resides in room \( r \), \( 1 \); otherwise, 0.
  - \( G_{ch} \) If task \( i \) on shift \( s \) belongs to a resident from cohort \( h \), \( 1 \); otherwise, 0.

- **Binary Variables:**
  - \( y_{cr} \) If SM \( c \) serves at least one of the residents of room \( r \) within the planning horizon, \( 1 \); otherwise, 0.
  - \( z_{ch} \) If SM \( c \) serves at least one of the residents of cohort \( h \) within the planning horizon, \( 1 \); otherwise, 0.

Now, we can formulate the model for pandemic conditions as below:

**Objective function (1)**

\[
\min F_2 = \sum_{c \in C} \sum_{r \in R} y_{cr} \quad (22)
\]

**Objective function (22) minimizes the number of distinct SMs serving each room.**

**Objective function (23)** minimizes the number of distinct cohorts served by each SM. The objective functions are restricted by the following constraints. Constraint sets (2), (3), and (5) - (21)

\[
\sum_{i \in T, c \in C} x_{ijcs} \leq A_{Ph} \quad \forall j \in T; c \in C; h \in H; s \in S; C_{jhs} = 1
\]

(24)

To consider the preference of SMs to work with different cohorts, we need to reformulate Constraint set (4). Therefore, Constraint set (24) assures that SMs are allocated to shifts and cohorts for which they are available to work.

\[
\sum_{i \in T, c \in C} x_{ijcs} + \sum_{i \in T, c \in C} x_{incs} \leq 1 \quad \forall j, n \in T; j > n; c \in C; h, m \in H; h \neq m; s \in S; C_{jhs} = 1; C_{jms} = 1
\]

(25)
Constraint set (25) guarantees that each SM is only allowed to serve one cohort of residents within a shift.

\[
\sum_{i \in I; j \in J} x_{ijcs} \leq y_{cr} \quad \forall j \in J; c \in C; r \in R; s \in S; V_{js} = 1
\]  

(26)

\[
\sum_{i \in I; j \in J} x_{ijcs} \leq z_{ch} \quad \forall j \in J; c \in C; h \in H; s \in S; G_{jhs} = 1
\]  

(27)

Constraint sets (26) and (27) are linking equations that connect \(x_{ijcs}\) with \(y_{cr}\) and \(z_{ch}\), respectively. Constraint set (26) forces \(y_{cr}\) to be equal to one if SM \(c\) serves at least one task for residents of room \(r\) during the planning horizon. Likewise, Constraint set (27) forces \(z_{ch}\) to be equal to one if SM \(c\) serves at least one task for a resident of cohort \(h\) throughout the planning horizon.

\[
y_{cr}, z_{ch} \in \{0, 1\} \quad \forall c \in C; r \in R; h \in H
\]  

(28)

Finally, Constraint set (28) specifies the new binary variables.

The above model for pandemic conditions has utilized an intuitive logic (i.e., Objective function (22)) to reduce the risk of spreading infectious diseases in RCFs. Moreover, we have incorporated the practical guideline of [59] (cohorting policy) and have added Objective function (23) and Constraint sets (24) and (25) into the mathematical model in order to facilitate RCF scheduling in pandemic conditions. While respecting such concerns, the proposed model aims to create RCF schedules with minimum costs.

3.3. Problem complexity

We can show the NP-hardness of our problem using a special case with the following simplified assumptions. Each task requires one SM, and an SM cannot do multiple tasks at the same time. SMs are identical. Time windows of tasks are relaxed, and the cohorting policy is not applied (i.e., any SM can handle any task). Objective function is also simplified to minimize the maximum overtime.

Considering the above relaxations, the staff scheduling problem for RC is reduced to a parallel machine scheduling problem (i.e., \(P_m | C_{max} \)), which is strongly NP-hard [58]. SMs correspond to identical machines and tasks with different lengths correspond to jobs with different processing times. As a result, the NP-hardness of the special case indicates that the general problem is also NP-hard.

4. Solution approach

Our preliminary experiments showed that the proposed MILP models (for non-pandemic and pandemic conditions) are efficient only on small-sized test instances. Accordingly, to deal with larger sized instances, this section presents a new and efficient heuristic algorithm for the staff scheduling problem in RC settings. Second, it provides a brief description about the benchmark metaheuristic algorithm adapted for our problem.

4.1. Heuristic algorithm

Optimization algorithms typically include a series of steps, each with a set of choices. Heuristic algorithms usually make locally optimal decisions with the hope of finding a globally optimal solution through exploration. Although these algorithms do not always find optimal solutions, they can often solve several instances to optimality [18]. Heuristics have three main advantages, (a) they are applicable and easily adaptable to a wide range of problems, (b) they generally have a simple structure and are easy to implement, and (c) these algorithms usually have a small set of parameters [66]. In this study, we design and develop a heuristic algorithm that starts with empty solutions and gradually constructs a population of feasible solutions. We choose a population-based algorithm since we want to: (a) increase the diversity of solutions and prevent local optimal solutions (by incorporating the randomness component into the algorithm - random decisions), and (b) to be able to create a non-dominated set of solutions. Then, to make the heuristic algorithm asymptotically optimal, we hybridize it with a local search method.

Solution representation

The heuristic we develop uses a cell structure with \(|C| \times |S|\) cells (\(|C|\) and \(|S|\) denote the numbers of SMs and shifts, respectively) to represent each solution, in which the assignment and scheduling of each SM-shift are represented by a \(2 \times \phi\) matrix (\(\phi\) refers to the number of tasks assigned to each SM-shift). The first and second rows of this matrix indicate the tasks allocated to an SM-shift and their starting times, respectively. For example, Fig. 1 provides an illustration of the solution representation for an instance with ten tasks, four SMs and one shift.

Main body

Before detailed discussions on different components of the heuristic algorithm, we briefly explain the main body of this algorithm (with six stages) in the following.

Stage 1: Initialize all parameters of the problem and the algorithm (including the population size - \(nPop\)) and create an empty solution

Stage 2: Assign tasks to SMs throughout the planning horizon by decomposing the problem based on the qualification levels of tasks

- Recognize all relevant tasks for each qualification level
- Randomly sort the tasks (\(T\)) and identify their rooms (\(R\))
- From tasks inside \(T\), group them based on their cohorts (without changing the order) and concatenate them such that tasks of each cohort appear in a row. This helps to decrease the movement of SMs between different cohorts (Fig. 2 illustrates an example with 11 tasks and three cohorts displaying how the algorithm groups randomly sorted tasks based on their cohorts)
- For each room \(j \in R\), apply an assignment procedure to allocate all tasks of the room to qualified SMs
- Restart to construct another solution if it is not possible to allocate all of the tasks

Stage 3: Schedule tasks based on their assignment

- For each SM-shift, detect allocated tasks, sort them based on their earliest preferred starting times and schedule them using a scheduling procedure
- Similar to Stage 2, restart to construct another solution if the overtime of the selected SM exceeds \(mO\) (the maximum allowable overtime)

Stage 4: Apply the repair procedure to those solutions that are infeasible with regard to synchronized tasks (allocated SMs to these tasks may not start serving them at the same time). Create another solution if the incumbent solution is infeasible after this stage.

Stage 5: Incorporate a local search method to increase the quality of the constructed solution by making a trade-off between violations from the earliest and latest preferred starting times of tasks

Stage 6: Compute the objective function values for the solution and terminate the search process if \(nPop\) feasible solutions were found

Once \(nPop\) feasible solutions are found, the algorithm removes duplicate solutions (solutions with the same objective function values and the same variable settings) and applies the non-dominated sorting technique. Proposed by Deb et al. [21], this technique finds non-dominated solutions (the Pareto front), where no solution is superior to others over all objective functions. To be more precise,
A vector of objective functions $F_i^*$ is on the Pareto front if there does not exist another solution such that $F_j^* \leq F_i^*$ for all $i \in [1, 2, 3]$ and $F_i^* < F_j^*$ for at least one $i$. If the termination criterion is not met, the algorithm stores the incumbent solution and restarts to construct another solution. Algorithm 1 in the supplementary material file provides a pseudo-code for the assignment procedure.

In the scheduling procedure used in Stage 3, the algorithm first retrieves the input parameters, including a list of tasks assigned to an intended SM-shift that is sorted based on their earliest preferred starting times ($T$). Then, the procedure sets the start time of the first task in $T$ equal to its earliest preferred starting time and eliminates this task from $T$. For each task $i$ in $T$, the algorithm identifies its earliest preferred starting time ($ET_i$), and schedules it to the earliest idle time of the SM greater than or equal to $ET_i$. This procedure may create schedules with allowable overtime (less than or equal to $mO$) or unallowable overtime (greater than $mO$). With probability $p_s$, the procedure attempts to decrease the unallowable overtime; with probability $1 - p_s$, it reduces the allowable overtime. For both cases, while there exists unallowable/allowable overtime and it is possible to reduce it, the procedure attempts to reschedule tasks (starting from the last one) to one time slot earlier. For example, if SM 1 in Fig. 1 experiences allowable overtime, with probability $1 - p_s$, the algorithm first tries to reschedule task T9 to one time slot earlier. This is continued until SM 1 does not face allowable overtime or it would not be possible to reschedule task T9 to a time slot earlier. If there is no free time slot before task T9 and after task T2 and SM 1 still experiences allowable overtime, the scheduling procedure looks to move both tasks T9 and T2 to one time slot earlier (the rest of the tasks are checked if the SM suffers from allowable overtime). Note that starting from the last task ensures the least movement of tasks, therefore, it may cause less violation from the time window constraint. Moreover, focusing sometimes on unallowable overtime (instead of always on allowable overtime) could be beneficial because reducing allowable

![Fig. 1. An illustration of the solution representation for the heuristic algorithm.](image1)

![Fig. 2. An illustration of grouping the randomly sorted tasks based on their cohorts.](image2)
over time might create significant violations for the time windows of tasks. At the end, if there is still unallowable overtime, the solution is called infeasible and the algorithm restarts to create another solution. Algorithm 3 in the supplementary material file provides a pseudo-code for the scheduling procedure.

Repair procedure

As mentioned previously, schedules created by the scheduling procedure may be infeasible with regard to synchronized tasks. Thus, the heuristic algorithm incorporates a procedure to repair such infeasibilities as follows:

- Recognize groups of synchronized SMs (C) on shift s, each group serving a common set of synchronized tasks (e.g., Fig. 4 illustrates a typical classification of six SMs and four tasks, where there are two groups of synchronized SMs).
- For each group of synchronized SMs C′ in C, recognize all synchronized tasks assigned to them (T) and sort them based on their earliest preferred starting times.
- Phase 1 - for each synchronized task i in T:
  - Identify SMs that are assigned to it
  - Schedule the task at the earliest idle time of all associated SMs, which must be greater than or equal to ET_{i0}
- Phase 2 - for each SM c′ in the group of synchronized SMs C′:
  - Recognize all non-synchronized tasks assigned to the SM (T′)
  - For each task j in T′, schedule the task at the earliest idle time of the SM (not overlapping with other already assigned tasks), which must be greater than or equal to ET_{j1}
- If constructed schedules have unallowable/allowable overtime:
  - (a) With probability pr, attempt to decrease the unallowable overtime, and (b) with probability 1 – pr, reduce the allowable overtime:
    - First, try to decrease the unallowable/allowable overtime for each SM c′ in C′ separately. For this reason, reschedule tasks after the last synchronized task of SM c′ to one time slot earlier until the unallowable/allowable overtime of this SM can be diminished
    - Second, if at least one of the SMs in c still has unallowable/allowable overtime, recognize time slots where all these SMs are idle (id). While id is non-empty and at least one of the SMs in C′ has unallowable/allowable overtime, reschedule all tasks scheduled after the last empty time slot in id to one time slot earlier and removes the last element of id
  - If there is still unallowable overtime, the solution is called infeasible and the algorithm restarts to create another solution.

The pseudo-code of this procedure can be found in the supplementary material file, Algorithm 4. Note that probabilities ps and pr are used in the scheduling and repair procedures, respectively, to increase the exploration capability of the heuristic algorithm. Local search method

According to our experimental observations, the heuristic algorithm usually schedules tasks after their earliest preferred starting times (no violation). However, it is more likely for the proposed heuristic algorithm to schedule tasks after their latest preferred

---

![Fig. 3. Priority of selecting SMs for serving task i associated with room r, cohort h and shift s.](image)

![Fig. 4. An illustrative example of the collection of synchronized SMs.](image)
starting times (violation). If the algorithm can make smart violations from the earliest preferred starting times as well, it might sometimes be able to prevent significant violations from the latest preferred starting times and overtime. For this purpose, we aim to develop a local search method in the following.

We explain the local search method through an example. Suppose an SM serves 11 non-synchronized tasks on one shift. The regular length of the shift and the maximum allowable overtime are equal to 32 and four time slots, respectively. Fig. 5 illustrates a sample of assignment and scheduling for this SM. Based on this figure, the SM is working two time slots in overtime, and the total violations from the earliest and latest preferred starting times are equal to zero and six time slots, respectively (viol_E and viol_t). The local search method attempts to recognize blocks of consecutive positive values in viol (showed by red rounded-rectangles) and stores the location of the first and last tasks of each block in B (a block is represented by bk_i). For this example, B is a matrix as follows:

\[
\begin{pmatrix}
  bk_1 & bk_2 & bk_3 & bk_4 \\
  3 & 6 & 8 & 11 \\
  3 & 6 & 9 & 11
\end{pmatrix}
\]

The local search method identifies all 1- and 2-combinations of blocks and stores them in CM as below (a combination is represented by cm_i).

\[
\begin{pmatrix}
  cm_1 & cm_2 & cm_3 & cm_4 & cm_5 & cm_6 & cm_7 & cm_8 & cm_9 & cm_{10} \\
  bk_1 & bk_2 & bk_3 & bk_4 & bk_5 & bk_6 & bk_7 & bk_8 & bk_9 & bk_{10}
\end{pmatrix}
\]

The algorithm randomly sorts columns of Matrix CM and starts to evaluate each combination in this matrix. For each combination cm_i, it looks for tasks before the first task of the combination and finds the location of the last task with a free time slot before it (lc). For the above example, the last task slot for both cm_1 and cm_2 is found before T4 (lc = 1). If a free time slot is not found, the local search method checks the rest of the combinations in CM; otherwise, it reschedules all tasks located in between lc and the last task of combination cm_i (inclusive) to one time slot earlier. Finally, the algorithm evaluates whether the new solution has a lower cost compared to the incumbent solution. If the new solution is better, the incumbent solution is replaced with the new one and the local search restarts again; otherwise, the algorithm continues to check other combinations in CM. The local search method terminates when there is no possible movement to improve the quality of the incumbent solution. The local search method is summarized in the supplementary material file, Algorithm 5. Note that the local search method explained above is only applicable to SMs serving non-synchronized tasks. For SMs that serve synchronized tasks, Algorithm 5 is adapted such that it reschedules synchronized SMs simultaneously (like decreasing unallowable/allowable overtime in the repair procedure).

4.2. Ant colony optimization

In the real world, ants wander randomly when looking for food. During this random search, they are attracted to a substance, called pheromone, left by other ants on their way back to the colony after finding food. The pheromone level of a path increases (or become more attractive) if several ants use it to reach food. At the same time, the pheromone level of a path decreases (or evaporates) as time passes. Inspired by this collaborative behavior of ants, we aim to adapt an Ant Colony Optimization (ACO) algorithm as a benchmark for the proposed heuristic algorithm. ACO algorithms are applied to solve a variety of discrete optimization problems [25]. Neto and Godinho Filho [56] and Deepalakshmi and Shankar [23] conclude that ACOs are hugely viable approach to solve scheduling problems, and Decerle et al. [22], Martin et al. [51], Zhou et al. [72] demonstrated their practicality specifically for staff scheduling problems.

Since the vehicle routing problem studied by Schyns [62] and our staff scheduling problem share similar features, such as consideration of time windows and heterogeneous fleets, the ACO of Schyns [62] can be adapted to solve our problem. Consideration of time windows and heterogeneous fleets are analogous to
Fig. 6. Flowchart of the proposed NA-ACO.

5. Computational results

In this section, we conduct a comprehensive numerical study in order to show the efficiency of the proposed solution approach and provide managerial implications. Accordingly, we first explain the generation of test instances and tuning parameters of the heuristic and the NA-ACO algorithms in Sub-sections 5.1 and 5.2, respec-
tively. Then, in Sub-Section 5.3, we show the efficacy of the heuristic algorithm’s components, including the repair procedure and the local search method. In Sub-Section 5.4, we compare the proposed heuristic algorithm with the mathematical modeling and the NA-ACO algorithm in terms of the solution quality and CPU time. Note that the maximum available CPU time for each run of the algorithms is set to 120 minutes. Since we are dealing with a multi-objective optimization problem, we will use IBM Ilog CPLEX and the epsilon constraint method (a posteriori method) for mathematical modeling in order to generate non-dominated solutions. Note that posteriori methods convey more information to the decision maker compared to priori methods. They give the whole picture (i.e., the Pareto set) to the decision maker before her final choice [52]. Starting from $F_1$, we solve the mathematical model based on this objective function in order to specify its lower bound ($F_1^u$) and use the found solution to determine possible upper bounds of other objective functions. Then, we repeat this procedure for the other two objective functions to find their lower bounds ($F_2^u$ and $F_3^u$) and other possible upper bounds. Finally, for each objective function, we find the upper bound ($F_2^u$, $F_2^u$, $F_3^u$) and set it to the maximum of its possible upper bounds. Once the lower and upper bounds are determined, we solve the mathematical model based on each objective function while considering the rest of them as constraints. For example, Constraints (29) and (30) are added to the model when we solve the problem according to $F_1$:

$$F_2 \leq F_2^u - \epsilon (F_2^u - F_2^l)$$  \hspace{1cm} (29)$$

$$F_3 \leq F_3^u - \epsilon (F_3^u - F_3^l)$$  \hspace{1cm} (30)$$

where $\epsilon$ is a constant and takes values in $[0,1]$. We start with $\epsilon = 0.2$ ($\epsilon = 0$ has already been used in determining the lower and upper bounds) and increase it by 0.2 step-size each time that we run the model. By repeating this procedure for all objective functions, we can finally find a set of non-dominated solutions. The CPU time here for each run is 120 minutes divided by 18 (the total number of runs), 400 seconds. Note that we do not choose a smaller step-size as it decreases the CPU time of each run significantly (CPU time of each run becomes equal to around 200 seconds when the step-size is 0.1). We also conduct some numerical analyses on different features of the problem in Sub-Section 5.5 to present managerial implications. The mathematical programme is coded in PyCharm 2020.3.5, in which the interpreter is Python 3.7, and is run on an Intel(R) Core(TM) i9-10900X CPU @ 3.70GHz with 64 GB of DDR4 RAM. While the heuristic and NA-ACO algorithms are coded in MATLAB 2020a and run on an Intel(R) Xeon(R) Platinum 8160 CPU @ 2.10GHz with 4 GB of DDR4 RAM.

5.1. Data

To ensure the realism of our model, we collaborated with a continuing care organization whose mission is to improve the care of seniors and vulnerable populations. However, because this research has been conducted during the COVID-19 pandemic, we could not find an available RCF that would provide data for our problem. Thus, we generate all instances in line with the guidelines of Public Health Ontario (e.g., Public Health Ontario [59]) and with the help of an expert from our partner organization. For two reasons, we generate two different sets of instances in this paper (called theoretical and realistic instances). First, the difficulty of instances has a strong positive correlation with the number of tasks for the proposed mathematical programme (it cannot even find an integer feasible solution for instances with 40 or more tasks in each shift). Therefore, for the sake of comparison, the maximum number of tasks required in shifts does not increase relative to the number of rooms in the theoretical instances, while the realistic ones have significantly more tasks in each shift. This allows the mathematical programme to find at least one integer feasible solution for the more theoretical instances which in turn enables us to better evaluate the performance of the heuristic algorithm. Second, it is evident that using two different sets of instances in the algorithm comparison increases the reliability and robustness of our analyses.

For the theoretical instances, we discretize the time into time slots of five minutes. A day consists of at most three shifts, each with a length of 96 time slots (eight hours) and 12 time slots of overtime (one hour). The planning horizon includes at least one day and at most seven days. We also assume that service times of tasks might vary between three to 18 time slots (i.e., 15 to 90 minutes). Tasks are associated with one to eight qualification levels (or types of SMs), including registered nurses, nurse practitioners, registered practical nurses, personal support workers, social workers, psychologists, and psychiatrists, where each type of SMs is able to serve a set of unique types of tasks. In line with Public Health Ontario [59], at most four cohorts of residents are considered for theoretical instances. As in Méndez-Fernández et al. [53], we assume that the length of the time window for each task could be at most equal to the regular length of the shift. In the theoretical instances, we set the maximum percentage of a shift’s regular length during which SMs can serve predefined tasks equal to 100%. The salary of part-time SMs per shift can take any value in {$\$120, $130, \ldots, $180$. The overtime cost of SMs per time slot can take any value in {$\$1, $1.5, \ldots, $4}. Finally, the penalty of violation from the time windows varies between $\$1-5 per time slot. Based on these settings, we generate 66 instances (Instances 1-66), each with ten different variants (560 instances in total). Note that variants of each instance would be different in terms of $P_{Ar}$, $P_{Ir}$, $P_{Br}$, $Q_{irs}$, $Q_{irs}$. $NC_{irs}$, $NC_{irs}$, $V_{irs}$, $G_{irs}$, $ET_{irs}$, $LT_{irs}$ and $AP_{irs}$. In Table 1, we show the configurations of the theoretical instances.

With respect to the realistic instances, each instance has two working shifts since predefined tasks mostly occur in the morning (7:00 to 15:00) and afternoo

...
Table 1
Configurations of the theoretical instances.

| Inst. | (|\mathcal{P}|, |\mathcal{S}|) | \#T | \#R | \#C | \#R’ | Number of qualification levels |
|-------|-------------------------|-----|-----|-----|-----|-------------------------------|
| 1, 12, 23, ..., 56 | (1, 1), (1, 2), (1, 3), (2, 6), (4, 12), (7, 21) | 5 | 1 | 8–18 | 10–20 | 1–2 |
| 2, 13, 24, ..., 57 | | 10 | 1 | 14–24 | 16–26 | 1–2 |
| 3, 14, 25, ..., 58 | | 15 | 1 | 18–36 | 26–36 | 1–2 |
| 4, 15, 26, ..., 59 | | 20 | 2 | 28–68 | 24–40 | 3–4 |
| 5, 16, 27, ..., 60 | | 25 | 2 | 40–92 | 28–48 | 3–4 |
| 6, 17, 28, ..., 61 | | 30 | 3 | 60–136 | 32–50 | 5–6 |
| 7, 18, 29, ..., 62 | | 40 | 3 | 70–148 | 48–66 | 5–6 |
| 8, 19, 30, ..., 63 | | 45 | 3 | 80–160 | 62–86 | 7–8 |
| 9, 20, 31, ..., 64 | | 50 | 4 | 100–200 | 78–100 | 7–8 |
| 10, 21, 32, ..., 65 | | 55 | 4 | 120–220 | 96–110 | 7–8 |
| 11, 22, 33, ..., 66 | | 60 | 4 | 140–240 | 106–120 | 7–8 |

Each row represents six instance types, where (|\mathcal{P}|, |\mathcal{S}|) is equal to (1, 1) and (7, 21) for the smallest and largest instances of each row, respectively.

Table 2
Configurations of the realistic instances.

| Inst. | (|\mathcal{P}|, |\mathcal{S}|) | \#T | \#R | \#C | \#R’ | Number of qualification levels |
|-------|-------------------------|-----|-----|-----|-----|-------------------------------|
| 67, 77, 87, ..., 107 | (1, 2), (2, 4), (3, 6), (5, 10), (7, 14) | 12 | 1 | 5–10 | 2 (1, 1, 1) | 1–3 |
| 68, 78, 88, ..., 108 | | 24 | 1 | 5–10 | 4 (2, 2, 2) | 1–3 |
| 69, 79, 89, ..., 109 | | 36 | 2 | 10–20 | 6 (3, 3, 3) | 1–3 |
| 70, 80, 90, ..., 110 | | 48 | 2 | 10–20 | 8 (4, 4, 4) | 1–3 |
| 71, 81, 91, ..., 111 | | 60 | 3 | 15–25 | 10 (5, 5, 5) | 1–3 |
| 72, 82, 92, ..., 112 | | 72 | 3 | 15–30 | 12 (6, 6, 6) | 1–3 |
| 73, 83, 93, ..., 113 | | 84 | 3 | 15–30 | 14 (7, 7, 7) | 1–3 |
| 74, 84, 94, ..., 114 | | 96 | 4 | 20–40 | 18 (9, 9, 9) | 1–3 |
| 75, 85, 95, ..., 115 | | 108 | 4 | 25–50 | 22 (11, 11, 11) | 1–3 |
| 76, 86, 96, ..., 116 | | 120 | 4 | 25–60 | 26 (13, 13, 13) | 1–3 |

Each row represents six instance types, where (|\mathcal{P}|, |\mathcal{S}|) is equal to (1, 1) and (7, 21) for the smallest and largest instances of each row, respectively.

Table 3
Potential values for the parameters of the heuristic algorithm and NA-ACO algorithm.

| Algorithm | Parameter | Description | Type | Range |
|-----------|-----------|-------------|------|-------|
| The heuristic algorithm | \(nPop\) | Size of population | Cardinal | (50, 100, 200, 400, 600, 800, 1000, 2000) |
| | \(ps\) | Probability of decreasing unallowable overtime in the scheduling procedure | Real | (0, 1) |
| | \(pr\) | Probability of decreasing unallowable overtime in the repair procedure | Real | (0, 1) |
| The NA-ACO algorithm | \(mI\) | Maximum number of iteration | Cardinal | (1, 2, 3, 5, 10, 20) |
| | \(mCyc\) | Maximum number of cycle | Cardinal | (5, 10, 20, 30, 40, 60) |
| | \(nAnt\) | Size of population | Cardinal | (10, 20, 30, 40, 60, 80) |
| | \(\beta\) | Heuristic exponential weight | Real | (0, 3) |
| | \(\rho\) | Exploration rate | Real | (0, 1) |
| | \(Q_0\) | Constant values for objective function | Real | (0, 1) |
| | \(Q_i\) | | Cardinal | (10, 50, 100, 200, 500, 1000, 1500, 2000) |

It should be noted that the set of realistic instances also includes small-sized instances that would not be realistic. We are forced to include such small instances in this set in order to be able to compare the performance of the heuristic algorithm over the MILP model. To save space, we have not provided all the information of theoretical and realistic instances here, and only stored them on the web. Thus, interested readers can use them for any future comparisons.

5.2. Tuning parameters

Both the heuristic and the NA-ACO algorithms include parameters whose values must be fully specified before use. The list of all parameters and their potential values are provided in Table 3. The choice of these parameters is significant for two main reasons. First, such algorithms are not parameter robust and might be inefficient with inappropriate parameter choices [28, 57], and second, random choices of parameters would lead to an unfair comparison of the two algorithms [71]. For this problem, we apply the iterated racing package proposed by Lópe-Zlópez et al. [47] for automatic algorithm configuration (available on R as the iRace package). We apply this package to the theoretical set of instances since it has a wide range of instances, each with ten variants. From the theoretical set, we use Instances 23–27 and 34–38 (we do not use larger size instances because the tuning process might take weeks, especially for the NA-ACO algorithm). To use this package for multi-objective optimization algorithms that generate a set of non-dominated solutions, López-López et al. [47] suggested the use of the hypervolume indicator as a mean of comparing the Pareto fronts. As a well-known, well-behaved and widely used metric [12,41], this indicator evaluates the percentage of the feasible region dominated by a Pareto front. Consequently, the larger the indicator, the higher the quality of the Pareto front (the hypervolume indicator can take values in [0, 1]).

To calculate the hypervolume indicator, we need to determine reference points of the objective functions for each instance (for

1. https://doi.org/10.17632/y8gg4hpt491
a minimization problem, an upper bound that is dominated by the entire feasible region, and a lower bound that dominates the whole feasible region). One recognized method to determine reference points is to ask experts to define them. This method is straightforward but has two main downsides [12]. First, inappropriately small or large reference points might lead to an unfair comparison of algorithms. Second, user-defined reference points are not capable of dealing with online problems (where the feasible region changes). To address these problems, we adapt the approach suggested by Cao et al. [12]. We find a Pareto front for each of the algorithms, add them into a pool, and then, detect the non-dominated ones. Note that we find the Pareto fronts according to preliminary and user-defined parameters of the algorithms (\( nPop = 200, ps = 0.5 \) and \( pr = 0.5 \) for the heuristic algorithm; \( mlit = 5, mCyc = 20, nAnt = 20, \beta = 1, \rho = 0.3, \phi_0 = 0.3 \) and \( Q_0 = 10 \) for the NA-ACO algorithm). Then, we detect the maximum and minimum values of the objective function from the pool of non-dominated solutions (\( max_i \) and \( min_i \), respectively). Finally, we use Eqs. (31) and (32) to specify the reference points.

\[
\begin{align*}
\text{min}_i R_i &= \min_i - \delta (\max_i - \min_i) \quad \forall i \in 1, 2, 3 \quad (31) \\
\text{max}_i R_i &= \max_i + \delta (\max_i - \min_i) \quad \forall i \in 1, 2, 3 \quad (32)
\end{align*}
\]

where \( \delta = 0.01 \) [12,41]. In addition to the objective functions, we consider the CPU time of the algorithms as the fourth performance metric in tuning parameters. We use Eqs. (31) and (32) in order to calculate the reference points of the CPU time as well, where \( \text{min}_i \) = 0 and \( \text{max}_i \) = 120 minutes, respectively. It should be noted that after specifying the reference points, the hypervolume indicator is calculated based on a built-in function in MATLAB. As the final step, we specify the termination criterion of the iRace package equal to 1000 runs for each of the algorithms.

After running the iRace package, the best configurations of the heuristic and the NA-ACO algorithms are found and reported in Table 4.

### 5.3. Efficacy of the repair procedure and the local search method

Before comparing the performance of the algorithms, we would like to show the efficacy of the heuristic algorithm components, such as the repair procedure and the local search method. For the repair procedure, we compare the performance of the heuristic algorithm with the Frifita and Masoumoudi [30] (alternative) approach, both of which aim to prevent infeasibilities of synchronized tasks. For a synchronized task, Frifita and Masoumoudi [30] suggest delaying the starting time of an SM for the task to the maximum starting time of all other SMs assigned to the same task. Synchronization constraints are usually difficult to handle due to the so-called interdependence problem [26]. An SM could share synchronized tasks with multiple SMSs, and there might be synchronized tasks that require multiple SMSs (even more than two). Under this condition, we believe that the alternative approach may not be efficient because it schedules synchronized tasks of one SM regardless of the others. To investigate this issue, we have removed the repair procedure from the heuristic algorithm and adapted the alternative approach. Then, we compare the performance of the heuristic algorithm with the repair procedure and with the alternative approach.

The summary results for this analysis are presented in Table 5 (a more detailed report can be found in the supplementary material file). With the same logic mentioned in Sub-section 5.2, we have used the theoretical instances for this analysis (and for the rest of the analyses, except comparison of the algorithms - Table 7) and only picked instances with two and four days (Entries 34 – 55, 220 instance-variants). Note that the descriptive statistics for the hypervolume indicator, gap of hypervolume indicator and CPU time in Table 5 (and in the rest of tables) are computed based on their average values for each instance (over all ten variants). Also, the average gap of hypervolume indicator for each instance are obtained in two steps. First, we compare the hypervolume indicator of each algorithm-instance-variant with the best hypervolume indicator associated with the same instance-variant in order to calculate the gap (repeat this step for all instance-variants). Second, we find the average gap of hypervolume indicator for each algorithm-instance.

According to Table 5, the proposed repair procedure significantly outperforms the alternative approach. The table demonstrates that the average CPU time over all instances is 5.18 minutes for the repair procedure, while this value increases to 85.87 minutes for the alternative approach. The table also shows that the maximum CPU time for the repair procedure is much smaller (22.96 minutes compared to 120 minutes). Furthermore, Table 5 showcases that the repair procedure was able to find at least one feasible solution for all variants, compared to almost half of variants (120/220 variants) with the alternative approach. The average gap of hypervolume indicator for the repair procedure is 0.14% over all instances, while this value is 5.18% for the alternative approach. All this demonstrates the superiority of the heuristic algorithm with the repair procedure.

Next, we investigate the efficiency of the proposed local search method. The summary results of this analysis are presented in Table 6 (a more detailed report can be found in the supplementary material file). Using a Paired Sample T-Test with 95% confidence, we did not find enough evidence against the assumption that the average CPU times of the algorithm with and without the local search method are insignificantly different (5.18 and 4.89 minutes on average, respectively). With regard to the solution quality, the algorithm with the local search method outperforms in the majority of instances in terms of the gaps found for the hypervolume indicator. The average gap of hypervolume indicator for the local search method is 0.14% over all instances, while this value without the local search method is 1.45%. This may not be an overall significant difference, but we have seen variants where the gap of hypervolume indicator for the algorithm without the local search method reaches 10% (e.g., variant 3 of Instance 54). Because the local search method does not considerably deteriorate the CPU time of the algorithm, we believe that it is beneficial to implement it in the heuristic algorithm.

### 5.4. Comparison of the solution approaches

In this sub-section, we solve the theoretical and realistic instances in order to compare the algorithms. Table 7 provides the summary results for the theoretical and realistic instances (more
detailed reports for the theoretical and realistic instances are presented in the supplementary material file).

Considering the theoretical instances, the average CPU time for the heuristic algorithm is 4.98 minutes over all instances. Moreover, the average CPU times of the mathematical programme and the NA-ACO algorithm are at least ten times higher (78.93 and 60.41 minutes, respectively). The average gaps of hypervolume indicator for the mathematical programme, the NA-ACO algorithm and the heuristic algorithm are equal to 48.18%, 56.95%, and 0.22%, respectively. A 95% Paired Sample T-Test failed to reject that the differences between the gaps of the mathematical programme and the NA-ACO algorithm are insignificant (instance-variants with missing values have been discarded). Furthermore, the maximum gap of hypervolume indicator is 100% for the mathematical programme (with a standard deviation of 45.37%), and 100% for the NA-ACO algorithm (with a standard deviation of 35.35%). In contrast, the maximum gap is meaningfully smaller for the heuristic algorithm (gap of 6.5% with a standard deviation of 0.94%). It is evident that the heuristic algorithm significantly outperforms the benchmark algorithms in terms of CPU time and solution quality. We also found that the heuristic algorithm found at least one integer feasible solution for all variants of all instances. However, the mathematical programme and the NA-ACO algorithm were unable to find at least one integer feasible solution for 66 and five instance-variants, respectively. Interested readers can find an illustrative example of Pareto frontiers found by the solution approaches in the supplementary material file.

With regard to the realistic instances, Table 7 demonstrates that the average CPU time for the heuristic algorithm is 3.41 minutes over all instance. In contrast, the average CPU times of the mathematical programme and the NA-ACO algorithm are around 30 and 20 times higher than the heuristic algorithm (104.67 and 63.62 minutes), respectively. Results also show that the NA-ACO and heuristic algorithms have found at least one integer feasible solution for all instances, while the mathematical programme has not found even a solution for the majority of instances (72% of instances). Thus, we do not include this solution approach in the following comparisons. The average gaps of hypervolume indicator for the NA-ACO algorithm and the heuristic algorithm are equal to 55.66%, and 0%, respectively. The maximum gap is 97% for the NA-ACO algorithm (with a standard deviation of 73.4%). In contrast, the maximum gap is meaningfully smaller for the heuristic algorithm (gap of 0% with a standard deviation of 0%). Thus, similar to the theoretical instances, the heuristic algorithm significantly outperforms the benchmark algorithms over the realistic instances in terms of both CPU time and solution quality.

5.5. Further analyses and discussions

In this sub-section, we aim to analyze two features of the problem under consideration - multiple objective functions, and forcing SMs to serve only one cohort within a shift. Table 8 outlines all cases investigated in this sub-section. As a reminder, Objective function (1) minimizes the costs associated with the salary of part-time SMs (TCS), SMs’ overtime and violation of time windows. Objective function (22) minimizes the number of distinct SMs serving each room (DSM), and Objective function (23) minimizes the number of distinct cohorts served by each SM (DCH). Note that the ob-
Table 8
Definitions of cases used for further analyses.

| Case # | Definition |
|--------|------------|
| Case 1 | The original problem with no modification |
| Case 2–4 | The original problem optimized based on only Objective function (1), (22) or (23), respectively |
| Case 5 | The original problem without Constraint set (25) |

To assess the importance of integrating multiple objectives into our solution approach, we compare Case 1 where the problem is optimized based on all objective functions (the original problem) with Cases 2–4 where the problem is optimized based on Objective functions (1), (22) or (23) separately, respectively. Fig. 7 compares Case 1 with Cases 2–4. This figure reveals that the objective functions are not in line with each other (i.e., optimization of one of them does not optimize the others), especially for Objective functions (22) and (23). Cases 2–4 have performed well only with regard to their associated performance metric. For example, Case 3 has maintained the lowest values of Objective function (22) over all instances, but at the same time, this case has the worst performance with respect to the other objective functions (around 1,800% and 125% worse performance on Objective functions (1) and (23) on average, respectively). In contrast, Case 1 has illustrated a fairly good performance across all objective functions. It should be noted that Fig. 7 shows a considerable drop in objective function values at Instance 45. This instance is the smallest instance type.
Fig. 8. Comparison of performance metrics under different optimization settings (Cases 1 and 5 refer to the optimization problem with/without Constraint set (25), respectively).

As shown in Fig. 8, in comparison to Case 1, the application of Case 5 considerably decreases the costs (58.96% on average), while exacerbating other performance metrics slightly (2.05% on average for the number of distinct SMs serving each room and 6.67% on average for the number of distinct cohorts served by each SM). In addition, it can be seen that the cost differences are greater for larger size instances. For a more in depth comparison, Fig. 9 provides an illustrative example for the third variant of Instance 40, day 1 and shift 3, comparing the assignment and scheduling of SMs with/without Constraint set (25). Note that, in this figure, synchronized tasks are identified with a black rectangle around them. Sixteen SMs are used to serve residents when this constraint set is used while the scheduling without the aforementioned constraint set uses only six. The example also illustrates the low utilization of SMs when the constraint set is used (around 25% vs 75%). Finally, Fig. 9 shows that SMs can suffer from unbalanced workloads when the constraint set is used. These issues can cause dissatisfaction,

with four days while Instance 44 is the largest instance type with two days (a similar drop can be found in the next figures).

The results therefore show that we can use a holistic approach and make a smart trade-off between conflicting performance metrics. This can be achieved without incurring significant costs or implementing a conservative or restrictive policy (e.g., the transition of existing facilities away from shared rooms suggested by Liu et al. [45]) that may hinder service delivery to residents during such unprecedented times.

Now, we analyze how the relaxation of Constraint set (25), which ensures that each SM is only allowed to serve one cohort of residents within a shift, impacts the solution. Such a policy might put pressure on RCFs given their limited resources, while at the same time, being useful to mitigate the circulation of COVID-19 in these facilities during outbreaks. We relax this constraint set, solve Instances 34 – 55 (Case 5) and compare the results with the original problem (Case 1) in Fig. 8.
decrease the quality and safety of their services and create unintended challenges for management. It is noteworthy that a similar trend can be found in other instances. Based on the above-mentioned discussion, we could derive two main takeaways. RCFs should only limit their SMs to visit only one cohort during a shift as an effective measure when dealing with an out of control situation to mitigate the circulation of infectious diseases [9,43,61]. Such a strategy might also be a beneficial for RCFs to reduce personal protective equipment and surveillance testing resources use for cohorts at low risk [4,46]. Such resources have limited supply [61] and are important in controlling the spread of COVID-19 [4].

In Canada, the first response to the COVID-19 outbreaks in RCFs was an abrupt restriction on visitors [29]. The residents, who had often struggled with loneliness before the COVID-19 pandemic, have been limited to their rooms without being able to visit their families/friends [67]. However, the cohorting policy enabled RCFs to relax the initial restriction of visitors such that they could visit their loved ones under special conditions [54]. This is good for both SMs and residents because the involvement of residents-family/friends has been shown to reduce SMs’ workload, and the mortality and infection rates of residents [33]. On the other hand, due to restrictions on SM movement between facilities [45], RCFs are struggling with a shortage of resources (e.g., in Canada, the military was called into some RCFs where some of the residents were left in unhealthy and unsanitary conditions) [29]. Considering their limited resources and budget, Constraint set (25) should be carefully implemented since it can put a lot of pressure on a facility and may hinder on-time, patient-centered, high-quality service delivery. Another approach is to use a more moderate policy for assigning SMs to different cohorts (a compromise between the consideration and no consideration of Constraint set (25)). For example, RCFs can let SMs visit no more than two similar cohorts within each shift.

6. Conclusions

The RC sector has suffered from a lack of funding and standard guidelines at the best of times, even without the additional complexities of a global pandemic [11]. Many deaths could have been prevented if enough attention was given to preparing the RC sector for COVID-19 (like the attention paid to preparing the hospital sector). This marginal status combined with the lack of efficient planning has highlighted significant inadequacies within the RC sector [7,29].

Our study incorporated advanced analytics techniques to improve the current staff scheduling methods for RC under pandemic conditions. Thus, we proposed a versatile and efficient heuristic algorithm that could quickly assign and schedule SMs in an RCF. Using two sets of (theoretical and realistic) instances, we conducted computational tests to evaluate the heuristic algorithm. With a maximum CPU time of 120 minutes, we demonstrated that the heuristic approach outperforms two well-known benchmark solution approaches, a mathematical programme and a NA-ACO algo.
rithm. For the theoretical instances, the average gaps of hypervolume indicator for the mathematical programme, the NA-ACO algorithm and the heuristic algorithm are equal to 48.18%, 56.95%, and 0.22%, respectively. At the same time, the average CPU time for the heuristic algorithm is at least ten times better than the other benchmark algorithms (4.98 minutes over all instance, on average). Note that the mathematical programme run up against the computational issues for instances with 40 or more tasks in each shift (it could not even find an integer feasible solution). Likewise, for the realistic instances, the heuristic algorithm significantly outperformed the benchmark algorithms in terms of both solution quality and CPU time.

Moreover, we conducted numerical analyses on two main features of the problem (multiple objective functions and forcing SMs to serve only one cohort within a shift) to derive managerial implications. First, the results showed that our holistic approach is capable of considering multiple stakeholders with conflicting interests. It can create smart trade-offs between their interests and find high-quality solutions with respect to all performance metrics. Second, we found that forcing an SM to serve only one cohort within a shift (Constraint set (25)) puts pressure on RCFs because this policy (a) requires more SMs to feasibly serve all residents, (b) creates unbalanced workloads for SMs, and (c) considerably decreases the utilization of SMs. However, using this policy (alongside Objective function (23) and Constraint set (24)) has the potential to be an effective measure to reduce the spread of COVID-19 in RCFs. Since RCFs are struggling with a shortage of resources, especially SMs, our finding is in line with Duan et al. [27] that asked the government of Canada to increase the staffing capacity of RCFs in order to make them able to deal with upcoming pandemics.

As a future research direction, interested researchers can study a many objective variant of our staff scheduling problem while considering other constraints, such as a precedence constraint (e.g., task i before task j) and a disjunction constraint (e.g., task i before/after task j). Our paper focused on operational-level decisions associated with the scheduling of SMs in RCFs. Thus, another future research direction could be the investigation of tactical-level decisions, such as the RC capacity planning problem under pandemic conditions.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Amirhossein Moosavi: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. Onur Ozturk: Conceptualization, Methodology, Investigation, Writing – review & editing, Supervision. Jonathan Patrick: Conceptualization, Methodology, Investigation, Writing – review & editing, Supervision.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/jomega.2022.102671

References

[1] Abadi MQT, Rahmani S, Sharifi A, Ahmadi M, Hissaga: designation and scheduling of nurses for taking care of covid-19 patients using novel method of hybrid salp swarm algorithm and genetic algorithm. Appl Soft Comput 2021; 108:107449. doi: 10.1016/j.asoc.2021.107449.

[2] Afshar A, Sharifi F, Jalali M. Non-dominated archiving multi-colony ant algorithm for multi-objective optimization: application to multi-purpose reservoir operation. Eng Optim 2009;41(4):313–25. doi: 10.1080/030521508022469414.

[3] Aslino R. Nearly half of known covid-19 deaths in Canada linked to long-term care homes: Tan. https://www.ctvnews.ca/canada/nearly-half-of-known-covid-19-deaths-in-canada-linked-to-long-term-care-homes-tan-1.4893419; 2020.

[4] Andrew M, Searle SD, McElhaney JE, McNeil SA, Clarke B, Rockwood K, Kelvyn DJ. Covid-19, frailty and long-term care: implications for policy and practice. The Journal of Infection in Developing Countries 2020;14(5):428–32. doi: 10.3852/jidc.13003.

[5] Ayyas OT, Ross AD, Scalon MC, Ayyas B. Short-term nurse schedule adjustments under dynamic patient demand. Journal of the Operational Research Society 2022;1:1–20. doi: 10.1080/01605682.20239566.

[6] Bagheri M, Devin AG, Ianzono A. An application of stochastic programming for nurse scheduling problem in real world hospital. Computers & Industrial Engineering 2016;96:192–200. doi: 10.1016/j.cie.2016.02.023.

[7] Béland D, Lecours A, Marchildon GP, Mou H, Oliff MR. Fiscal federalism and equalization policy in canada: political and economic dimensions. University of Toronto Press; 2017.

[8] Van den Bergh J, Beljen J, De Brucker P, Demeulemeester E, De Boeck L. Personal scheduling: a literature review. Eur J Oper Res 2012;226(3):367–85. doi: 10.1016/j.ejor.2011.12.023.

[9] Brown KA, Jones A, Daneman N, Chan AK, Schwartz KL, Garber GE, Costa AP, Stall NM. Association between nursing home crowding and covid-19 infection and mortality in canada. JAMA Intern Med 2020;181(2):239–36. doi: 10.1001/jamainternmed.2020.6566.

[10] Canfield A, Morgan D, Stanisz W, McKitton K, Rycroft-Malone J, Dopson S, Estabrooks C. The hidden complexity of long-term care: how context mediates knowledge translation and use of best practices. Gerontologist 2005;45(4):1013–22. doi: 10.1093/geront/056.

[11] Canadian Healthcare Association. New directions for facility-based long-term care.cbc news. here a list of quebec long-term care homes with confirmed covid-19 cases. https://www.cbc.ca/15535245; 2009.

[12] Cao Y, Smucker BJ, Robinson TJ. On using the hypervolume indicator to compare pareto fronts: applications to multi-criteria optimal experimental design. J Stat Plan Inference 2015;160:60–74. doi: 10.1016/j.jspi.2014.12.004.

[13] Capanerea P, Scutellà MG, Nervi F, Galli L. Demand uncertainty in robust home care optimization. Omega (Westeport) 2018;80:95–110. doi: 10.1016/j.omega.2017.08.012.

[14] Centers for Disease Control. How covid-19 spreads. https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/how-covid-spreads.html; 2020.

[15] Chukwu A, Penny S. Natural language processing and social media in global healthcare. J Med Internet Res 2020;22(2):e18661. doi: 10.2196/18661.

[16] Cinar A, Salmon FS, Bokuya B. Prioritized single nurse routing and scheduling for home healthcare services. Eur J Oper Res 2021;298(3):867–78. doi: 10.1016/j.ejor.2019.07.009.

[17] Comas-Herrera A, Zalakain J, Litwin C, Husi AT, Lane N, Fernández JL. Mortality associated with covid-19 outbreaks in care homes: early international evidence. Article in LITCovid org, International Long-Term Care Policy Network, CPEC-LSE 2020-26.

[18] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. Cambridge: MIT press; 2009.

[19] Dantzig GB. Letter to the editor comment on edie’s attrative delays at tall booth. Journal of the Operations Research Society of America 1954;3(3):39–41. doi: 10.1287/opre.2.3.339.

[20] De Simone B, Chouillard E, Di Saverio S, Pagani L, Sertoli M, Biff W, Coccolini F, Prier A, Khan M, Borzelino G, et al. Emergency surgery during the covid-19 pandemic: what you need to know for practice. The Annals of The Royal College of Surgeons of England 2020;102(5):323–32. doi: 10.1308/rcsann.2020.0033.

[21] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002;6(2):182–97. doi: 10.1109/4235.996017.

[22] Decerle J, Grunder O, El Hassani AH, Barakat O. A hybrid memetic-ant colony optimization algorithm for the home health care problem with time window, synchronization and working time balancing. Swarm Intell 2019;13:171–83. doi: 10.1007/s11721-019-00209.

[23] Deshmukh R, Anakshi P, Shankar K. Role and impacts of ant colony optimization in job shop scheduling problems: a detailed analysis. Evolutionary Computation in Scheduling 2020;11:1–35. doi: 10.1007/s10667915429.ch2.

[24] Demirbilek M, Branke J, Strauss AK. Home healthcare routing and scheduling of multiple nurses in a dynamic environment. Flexible Services and Manufacturing Journal 2019;1:1–28. doi: 10.1007/s10696-019-09350-x.

[25] Dorigo M, Caro GD, Gambardella LM. Ant algorithms for discrete optimization. Ann Math 1999;5(2):137–72. doi: 10.1016/10646966958728.

[26] Dows-Costa A. Synchronization in vehicle routing survey with multiple synchronization constraints. Transportation Science 2012;46(3):297–316. doi: 10.1287/trsc.1104.0400.

[27] Duan Y, Iaconi A, Song Y, Norton PG, Squires JE, Kefee J, Cummings GG, Estabrooks CA. Care aides working multiple jobs: implications for staffing policies in long-term care homes during and after the covid-19 pandemic. J Am Med Dir Assoc 2020. doi: 10.1016/j.jamda.2020.07.036.

[28] Erfani B, Ebrahimnejad S, Moosavi A. An integrated dynamic facility layout and job shop scheduling problem: a hybrid nsga-ii and local search al-
A. Moosavi, O. Ozturk and J. Patrick Omega 112 (2022) 102671