JOURNAL OF DECISION MAKING AND HEALTHCARE
Volume 1 (2024), No. 2, 57–68
https://doi.org/10.69829/jdmh-024-0102-ta01
Tulipa Opera Scholarum

ANT COLONY SYSTEM FOR EFFECTIVE DOCTOR ROSTERING

CHUN-TE CHEN[1] AND MATTHEW M. LIN[1,*]

[1] *Department of Mathematics, National Cheng Kung University, Tainan, 701, Taiwan*

Abstract. Designing effective schedules for medical staff is vital in the healthcare industry as it directly impacts the efficiency of medical institutions, patient care, and staff satisfaction. To tackle this challenge, we suggest utilizing a binary programming model in combination with the ant colony system (ACS) approach. We have tailored the ACS method to address the constraints and requirements of medical staff scheduling, including adaptations to the pheromone matrix and the heuristic information. Numerical results illustrate our method's efficiency in generating optimal schedules, and we analyze how variations in ACS parameters affect the search process. These experiments highlight the robustness of our approach and pave the way for a promising future of further theoretical analysis and algorithm refinement.

Keywords. Ant colony system; Doctor rostering problem; Health screening serving; Scheduling.

1. Introduction

The doctor rostering problem, a variation of the nurse rostering problem, is recognized as one of the NP-hard combinatorial problems [14] and has been extensively studied over the past few decades [4, 18]. This problem involves scheduling doctors while accounting for their diverse skills and clinic days. Unlike the conventional nurse rostering problem, not all doctors can be assigned to every shift, adding complexity and making the problem more challenging. Additionally, we need to consider doctors' preferences and well-being to maintain a balance between their professional and personal lives. This study addresses a scheduling issue based on a real-life scenario in a hospital in Taiwan.

Historically, doctor shift scheduling was typically carried out manually by hospital administrators who relied on their experience and expertise. With the advancement of computer technology, various automation tools and optimization algorithms can be developed to address these challenges. However, due to the complexity of these problems, techniques such as metaheuristics such as simulated annealing [15, 23], genetic algorithms [16, 18], and Tabu search [3, 17] are often the better choices to find solutions effectively. Our work focuses on applying the ant colony system (ACS) method to address our specific problem rather than exploring various approaches.

However, the effectiveness of a metaheuristic approach highly depends on the size and characteristics of the problem [2]. Notably, in the ACS, two critical factors influence the behavior of the search method: the design of the heuristic [1], which balances against pheromone, and the tuning of the parameters [24]. Research has shown that these factors affect the convergence to the best solution and the required time. In our paper, we investigate the influence of these factors and shift our focus away from maximizing doctors' preferences, as done in [13] and [25]. Instead, we concentrate on predefined target shifts and integrate doctors' preferences as positive and negative lists within the framework of hard constraints. Consequently, our heuristic design revolves around minimizing deviations from the target shifts. Numerical results show that our approach effectively avoids local minima, a common issue for heuristic algorithms and that it demonstrates its efficiency.

*Corresponding author.
E-mail address: chchte0102@gmail.com (C.-T. Chen), mhlin@mail.ncku.edu.tw (M.M. Lin)
Accepted: August 23, 2024.

This paper is organized as follows. In Section 2, we define the problem we aim to resolve and build its corresponding binary mathematical model. We introduce the ACS method and explain the design of our strategy in Section 3. Numerical results are provided in Section 4, and the concluding remarks are given in Section 5.

2. Mathematical Model

Traditional nurse scheduling problems (NSP) usually deal with fewer employee categories and more straightforward shift types, like morning, day, and night shifts, assigned to a specific group of nurses. The primary goal is to minimize employee dissatisfaction with assigned schedules or maximize their preferences [13, 22]. This can be mathematically represented as:

$$\min \sum_i \sum_j c_{ij} X_{ij}$$

where c_{ij} represents a score indicating how much nurse i dislikes schedule j, and $X_{ij} = 1$ if schedule j is assigned to nurse i, and 0 otherwise. Various constraints, such as restrictions on consecutive work and night shifts, may apply depending on the hospital.

The doctor rostering problem is similar to the nurse scheduling problem but involves additional complexities. On the one hand, similar to the nurse scheduling problem mentioned earlier, the hospital's services are categorized into morning, afternoon, all-day, and on-call shifts. A doctor can handle only one task per time slot, meaning they cannot hold two tasks simultaneously in the morning or both morning and on-call shifts simultaneously. However, a doctor can take morning and afternoon shifts on the same day. On the other hand, our specific challenge is scheduling healthcare services for doctors in a hospital in Taiwan. In addition to their regular outpatient duties, each doctor is assigned various medical tasks within the hospital, such as cancer screening and palliative care, each of which requires the attendance of one doctor. Therefore, when we assign tasks, we must consider the doctors' outpatient schedules.

Below, we have listed the requirements that frequently need to be met:

1. **Shift Coverage**: Each service can assign exactly one doctor per available day, but it is also possible not to schedule anyone.
2. **Negative List**: Doctors are unavailable on specific days for their service, such as weekends or during annual leave.
3. **Positive List**: Doctors prefer to take certain services on specific days.
4. **Closing Days**: There are certain days when the service must be closed.
5. **Time Conflict**: Doctors in certain services cannot work on specific days due to time slot conflicts; for example, a doctor cannot be assigned to both a morning service and an on-call service on the same day.
6. **Clinic Conflict**: If a doctor has a clinic service, they cannot be assigned to another service during the same time slot on the same day.
7. **Total Shifts**: Each doctor has a lower and upper bound for the number of shifts they can be assigned to each service.

We list and interpret the required notations for our mathematical model in Table 1. We then use the notations to describe the problem of minimizing the total difference between the number of shifts assigned to each doctor and their target shifts as specified by the chief resident doctors at the beginning of the month as follows:

Objective

$$\min \quad f(X) = \sum_{s \in S} \sum_{m \in M_s} \left| \sum_{d \in D} x_{mds} - t_{m,s} \right| \tag{2.1}$$

TABLE 1. Notations for the Mathematical Model

Notation	Description
M	The set of all doctors
D	The set of all arranging days
S	The set of all medical service
M_s	The set of doctors responsible for service $s, \forall s \in S$
$S_{mor}, S_{noon}, S_{all}, S_{call}$	Sets of services in the morning, at noon, all-day, and on-call, respectively
N_m	$\{(d,s) \mid d \in D, s \in S\}$: Negative list for doctor $m, \forall m \in M$
P_m	$\{(d,s) \mid d \in D, s \in S\}$: Positive list for doctor $m, \forall m \in M$
C_s	$\{d \in D\}$: The days that service s is closed, $\forall s \in S$
$K_{m,mor}$	Doctor m has clinic in the morning and $K_{m,mor} \subseteq D$
$K_{m,noon}$	Doctor m has clinic at noon and $K_{m,noon} \subseteq D$
$V_{d,mor}$	$\{(d,s) \mid s \in S_{mor} \cup S_{all} \cup S_{call}\}$: The days that the clinic overlaps in the morning
$V_{d,noon}$	$\{(d,s) \mid s \in S_{noon} \cup S_{all} \cup S_{call}\}$: The days that the clinic overlaps at noon
$t_{m,s}$	Target shifts for each doctor, $\forall s \in S, \forall m \in M$
$U_{(m,s)}, L_{(m,s)}$	Upper and lower limits of service s for doctor m
Variable	**Description**
x_{mds}	Binary variable, equal to 1 if member m on day d has service s; otherwise, 0

Constraints

$$x_{mds} \in 0, 1, \quad \forall m \in M, \forall d \in D, \forall s \in S \tag{2.2a}$$

$$\sum_{m \in M} x_{mds} \leq 1, \quad \forall d \in D, \forall s \in S \tag{2.2b}$$

$$\sum_{m \in M} \sum_{(d,s) \in N_m} x_{mds} = 0 \tag{2.2c}$$

$$\sum_{(d,s) \in P_m} x_{mds} = |P_m|, \quad \forall m \in M \tag{2.2d}$$

$$\sum_{s \in S} \sum_{d \in C_s} x_{mds} = 0, \quad \forall m \in M \tag{2.2e}$$

$$\sum_{s \in S_{mor} \cup S_{all} \cup S_{call}} x_{mds} \leq 1, \quad \forall m \in M, \forall d \in D \tag{2.2f}$$

$$\sum_{s \in S_{noon} \cup S_{all} \cup S_{call}} x_{mds} \leq 1, \quad \forall m \in M, \forall d \in D \tag{2.2g}$$

$$\sum_{d \in K_{m,mor}} \sum_{(d,s) \in V_{d,mor}} x_{mds} = 0, \quad \forall m \in M \tag{2.2h}$$

$$\sum_{d \in K_{m,noon}} \sum_{(d,s) \in V_{d,noon}} x_{mds} = 0, \quad \forall m \in M \tag{2.2i}$$

$$L_{(m,s)} \leq \sum_{d \in D} x_{mds} \leq U_{(m,s)}, \quad \forall m \in M, \forall s \in S \tag{2.2j}$$

The constraints can be interpreted as follows. Constraint 2.2b ensures that at most one doctor can be assigned to the service each day. Constraint 2.2c ensures that no doctor is assigned to a day they cannot work, while Constraint 2.2d ensures that every doctor is assigned to the days they prefer. Constraint 2.2e ensures that no doctor is assigned on a day when the service is not available, such as weekends. Constraints 2.2f and 2.2g ensure there are no time conflicts. Constraints 2.2h and 2.2i ensure that a doctor will not have clinic service and medical service at the same time slot. Constraint 2.2j ensures each doctor's shifts fall within their respective lower and upper bounds.

3. Method

3.1. **The ant colony system method: a revisit.** The Ant Colony System (ACS) method, first proposed in [6], is inspired by the way ants search for food between their colony and a food source [5, 10]. As ants search, they leave pheromone trails that affect their decisions based on pheromone concentration and other factors like distance or energy. This process helps the colony find the most efficient path between their nest and the food source. Theoretically, the ACS method involves two main mechanisms: route selection and pheromone update [7, 8, 9]. Here, we explain these two mechanisms as follows.

The first mechanism, the route selection rule, probabilistically chooses paths based on pheromone values and heuristic information. If ant x is at node i, the rule for choosing the next node j is described as:

$$j = \begin{cases} \arg\max\limits_{n \in N_x(i)} \{[\tau(i,n)][\eta(i,n)]^\beta\}, & q \leq q_0, \\ J, & \text{otherwise}, \end{cases} \tag{3.1}$$

where $\tau(i,n)$ and $\eta(i,n)$ denote respectively the values of pheromone and heuristic information between nodes i and n, q is randomly selected from $[0,1]$ during the process, q_0 is the pre-determined threshold for balancing exploitation and exploration, $N_x(i)$ is the set of the nodes that the ant x can choose from at node i, $\beta > 0$ is the parameter for determining the relative importance of pheromone versus heuristic information, and J is a random variable denoting the available points $j \in N_x(i)$ whose probability distribution $P_x(i,j)$ is given by

$$P_x(i,j) = \frac{[\tau(i,j)][\eta(i,j)]^\beta}{\sum_{n \in N_x(i)} [\tau(i,n)][\eta(i,n)]^\beta}. \tag{3.2}$$

Specifically, at each decision point during the search process, an ant uses a randomly generated value q to determine whether to choose the best-known high-quality path (i.e., paths with higher $[\tau(i,n)]$ $[\eta(i,n)]^\beta$ value) or to try a new path according to (3.2). On the one hand, (3.1) describes the "exploitation" mechanism. Exploitation involves ants selecting paths already identified as high-quality, which can quickly lead to better solutions. This process means that when a particular path is more attractive to the ants, they will only choose it. On the other hand, (3.2) explains the process of "exploration." This mechanism involves ants trying new paths during the search process, even if these paths have lower pheromone concentrations. This determination increases the diversity of choices among ants, enhancing the chances of discovering better paths. Thus, ants are more likely to find other solutions that may be more optimal.

The second mechanism, the pheromone updating rule, is a technique that results in improved solutions. Similar to how real ants leave more pheromones on shorter paths, this step ensures that superior paths have higher pheromone values. Once combined with the preceding path selection rules (3.1) and (3.2), this increases the likelihood of ants choosing better paths. In the updating procedure, there is also

a mechanism to balance exploration and exploitation, known as the local and global updating rules, respectively. The following are the formulas for both rules. The first one is the *local updating rule*

$$\tau^{t+1}(i,j) = (1-\rho)\cdot\tau^{t}(i,j) + \rho\cdot\tau_0 \tag{3.3}$$

and the second one is the *global updating rule*

$$\tau^{t+1}(i,j) = (1-\alpha)\cdot\tau^{t}(i,j) + \alpha\cdot(L_{best}+\epsilon)^{-1}, \tag{3.4}$$

where $\rho \in [0,1]$ and $\alpha \in [0,1]$ respectively denote the local and global evaporation rate of the pheromone, τ_0 is the initial pheromone value, L_{best} is the best objective value, and $\epsilon > 0$ is a small selected number used to make (3.4) well-defined in case where the value of $L_{best} = 0$. That is, when an ant moves from point i to point j, the pheromone value is updated according to this local updating rule (3.3), This method prevents ants from continuously searching around local extrema, thus ensuring the mechanism of exploration. Once all ants complete their search, we use (3.4) to reinforce the pheromone on the route that leads to the best performance among the ants. This encourages ants to exploit the route that indicates a better solution.

3.2. **Configuration for doctor rostering.** To address our doctor scheduling issue using the ACS method, the initial inclination might be to apply the approach used for solving the Traveling Salesman Problem (TSP) [20]. However, the scenario differs significantly, and we elaborate on these distinctions below.

In our approach, we treat all selections as cities, allowing ants to choose these cities to generate feasible solutions. Specifically, ants first select the "service city," which represents the overall service that needs to be provided, followed by the "doctor cities," which correspond to each specific service. This means that each doctor city represents a unique service that a doctor can provide. If all doctors have already completed their target shifts, scheduling them further to fill the remaining unscheduled days in the month would lead to over-assignment. This would cause doctors to exceed their target shifts, resulting in a higher objective function value. Conversely, if all suitable doctors have reached their maximum allowable shifts, we will opt not to assign additional shifts and instead close the service. After excluding dates that must be removed in advance (for example, weekends or specified non-working days), we use the order of selection to determine how doctors are scheduled for the remaining dates. Unlike the traditional TSP [11], in our method, each city may be revisited, and ants may stay in the same city. This means that a doctor can be chosen multiple times or even scheduled for consecutive days. Therefore, in addition to considering pheromone values between any two doctor cities, we also record a pheromone value for a doctor selecting themselves.

Our algorithm begins by generating an initial solution to ensure that the feasible solution set is nonempty, which provides a foundation for updating the initial pheromone values. See also [2] for a similar procedure. We start by randomly selecting a service. We sequentially choose a doctor from the available options for the dates that have yet to be assigned for that service and note the selection order. Priority is given to doctors who have yet to reach their minimum shifts. Once all days are scheduled, we select the next unassigned service. After planning all services, we obtain an initial solution, X_0. During the exploration phase, we initialize the pheromone values. The initial pheromone value for selecting service i is denoted $\tau_s(0,i)$, and the pheromone value between services i and j is denoted $\tau_s(i,j)$, where $i \neq j$. To schedule service m and select doctor a, the pheromone is assigned as $\tau_m(0,a)$, and the pheromone value between doctor a and doctor b is denoted as $\tau_m(a,b)$, where a could be equal to b. The initial pheromone values are updated to

$$\tau_s(i,j) = \tau_m(a,b) = \tau_0 = (N\cdot f(X_0)+\epsilon)^{-1} \tag{3.5}$$

for all services i and j (with i possibly denoting 0 for the initial choice), and for all services m and doctors a and b, where a can also be 0 denoting the initial choice. Here, N is the total number of nodes,

and ϵ is a small number to ensure the well-defined nature of (3.5) when $f(X_0) = 0$, i.e., when the arranged shifts match the target shifts exactly.

In applications, the heuristic information $\eta(i, j)$ typically represents the reciprocal of the cost from node i to node j [10, 12]. For example, in the TSP [21], the heuristic is determined by

$$\eta(i, j) = \frac{1}{d_{ij}}$$

where d_{ij} denotes the distance between cities i and j. Other heuristic designs, such as those discussed in [25], include static and dynamic heuristics. However, these may not be universally applicable. The static and dynamic designs often involve selecting from pre-existing schedules and aiming to meet staff preferences. Since our scheduling is created from scratch without predetermined options, such heuristics are not suitable for our case. In our approach, we set $\eta_b(i, j) = 1$ between any two "service cities," as there is no apparent cost involved. For choosing doctor b from doctor a when arranging service m, the heuristic is defined as follows. Firstly, for each available doctor, we calculate the error err as the difference between their current shift and their target shift, i.e., $err =$ (current shift of $b -$ target shift of b). We then use the exponential function to compute the cost:

$$\eta_m(a, b) = \frac{1}{10000}(e^{err})^{-1} \tag{3.6}$$

The factor $\frac{1}{10000}$ aligns the magnitudes of τ and η for our later experiments and is adjustable based on different scenarios. This heuristic information prioritizes doctors whose assigned shifts are significantly below their target and imposes higher costs for assigning shifts to those who are close to or have exceeded their target. In other words, the cost decreases progressively as the number of already scheduled shifts decreases significantly and deviates further from the target shift. This makes choosing between two different doctors easier if there is a significant difference in the number of scheduled shifts.

Once we have set the pheromone and heuristic values, we can use the ACS method to solve the doctor rostering problem. This involves repetitively following the entire procedure until specific conditions are achieved, such as creating an optimal or satisfactory schedule. The detailed steps for using the ACS method to solve the doctor rostering problem are outlined in Algorithm 1.

4. Computational Results

The testing data is outlined as follows: we have 50 doctors available, a span of 30 days, and 15 services to organize. To replicate real-world scenarios and accommodate the specific needs of each service, we randomly assign lists of available doctors for each service, each containing 3 to 6 doctors. Each doctor can be assigned to up to three services. Additionally, each doctor has designated clinic days, with morning, afternoon, or all-day shifts ranging from 10 to 20 days. Doctors have positive and negative availability lists, specifying preferred and unavailable days, ranging from 2 to 6 days. Each doctor's minimum and maximum work shifts range from 1 to 7. Furthermore, we randomly select certain days when the business is not operational, such as weekends or other specified dates.

We evaluate the effectiveness of our scheduling method by ensuring that it not only meets strict constraints but also minimizes overall deviation from the targeted shifts assigned to each doctor for each service. Our testing data aims to maintain consistency in the number of shifts assigned to each doctor using the average number of shifts for each business as the target. However, in practice, the target shifts are determined by chief residents at the beginning of each month based on the anticipated patient load and other factors. We conduct five experiments to evaluate our method and provide a detailed discussion of each experimental result.

Algorithm Ant colony system method for doctor rostering problem

```
1: Construct initial solution
2: Set initial pheromone values
3: for each iteration do
4:     for each ant do
5:         while not every service is assigned do
6:             Choose a non-selected service by using the selection rules (3.1) and (3.2).
7:             Update the pheromone matrix of the service using the local updating rule (3.3).
8:             Arrange the day in positive list and closed days.
9:             for each non-arranged day do
10:                Choose a doctor using the selection rules (3.1) and (3.2).
11:            end for
12:            Update the pheromone matrix of the doctor by the selection order using (3.3).
13:        end while
14:    end for
15:    Select the best arrangement obtained from line 4 to line 14.
16:    if the best arrangement is better than the global best arrangement then
17:        Update the global best arrangement
18:    end if
19:    Update the pheromone matrix with the global best arrangement using the global updating
       rule (3.4)
20: end for
```

4.1. **Performance evaluation.** The first experiment aims to assess the performance of the ACS method using 40 independent testing datasets. We use two methods in each dataset to find the optimal objective values: the randomly selected method and the ACS method. We set the number of iterations for the randomly selected method to 20,000. Correspondingly, we set the number of iterations to 200 and the number of ants for each iteration to 100. Table 2 provides the ACS parameter settings and descriptions.

TABLE 2. Parameter settings for evaluating ACS performance

Parameters	Descriptions	Values
β	relative importance	1
ρ	local evaporation rate	0.3
α	global evaporation rate	0.3
ϵ	perturbation	0.01
q_0	threshold	0.7

In Figure 1, the vertical axis represents the deviation of all doctors' shifts from their target counts. A lower value indicates that the scheduled shifts of all doctors are closer to their target counts. In comparing the best objective value for each set of testing data, we noticed that, despite both methods having the same number of feasible solutions, our proposed method consistently outperformed the randomly selected method across the 40 different testing datasets. However, strict requirements can arise due to the random generation of testing data. For example, doctors may have negative lists or limited availability for coverage. Consequently, the optimal objective function value can vary significantly, from 10 to 47, indicating the challenge of initially scheduling to meet all doctors' needs.

Our upcoming experiments will involve the random selection of a specific set of testing data for further testing and parameter fine-tuning. Each experiment will be repeated 10 times, and we will

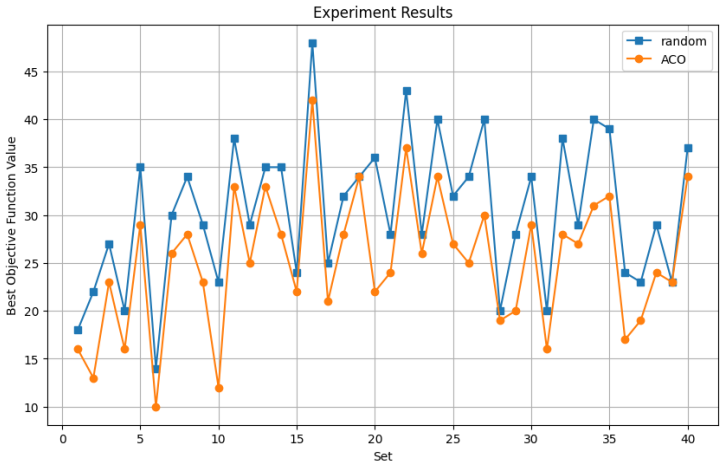

FIGURE 1. Experimental results for 40 testing datasets

calculate the average of the best objective values for each round to minimize the influence of rare lucky outcomes on the experimental findings. The ants will implement our suggested method according to the steps outlined in Algorithm 1.

4.2. **Impact of β.** It is well-known that configuring the factor β is essential in the ACS method [19]. It balances the influence of pheromone trails with the ants' search for optimal routes. Our study explores small and large values of β to assess their impact on experimental outcomes. To better distinguish the effects of parameter changes, we select β from the set $\{0, 0.01, 0.2, 1, 8\}$ instead of using uniform intervals. For each β value, we computed the objective values over 100 iterations with 100 ants for each iteration. At the same time, the other parameters were kept constant as specified in Table 3.

TABLE 3. Parameter settings for evaluating the impact of β

Parameters	Values
ρ	0.3
α	0.3
ϵ	0.01
q_0	0.7

In Figure 2, the results reveal minimal difference among the optimal outcomes for non-zero β values, suggesting that all have approached the optimal solution within 100 iterations. Additionally, the figure shows that higher β values give rise to faster convergence to the optimal solution. This indicates that, within our heuristic framework (3.6), prioritizing doctors with the most significant deviation from their target shifts generally yields better scheduling results.

Interestingly, even a tiny β value, such as $\beta = 0.01$, demonstrates a significant performance improvement over $\beta = 0$ despite being considerably smaller than 1. This improvement highlights the importance of the heuristic η. The heuristic information, denoted by η, offers valuable local problem knowledge or rules that help guide the ants more effectively through the search space. Without such heuristic guidance, the algorithm relies solely on random exploration, which significantly hampers convergence speed.

4.3. **Impact of threshold q_0.** In the Ant Colony System (ACS), finding a balance between investigating new possibilities and utilizing known solutions is important. This involves adjusting the parameters in

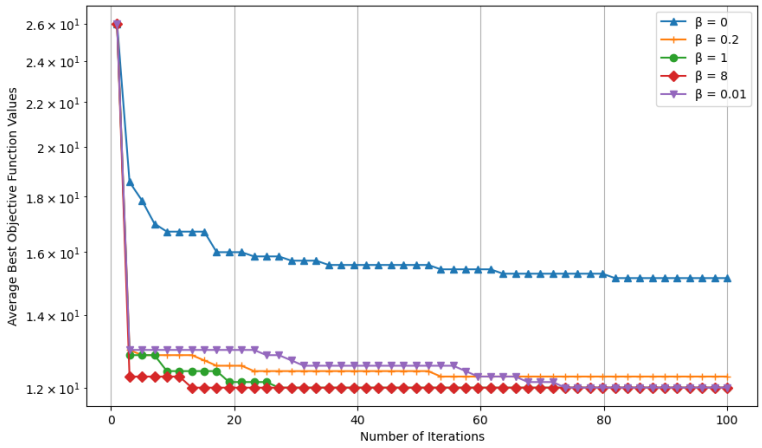

FIGURE 2. Impact of β

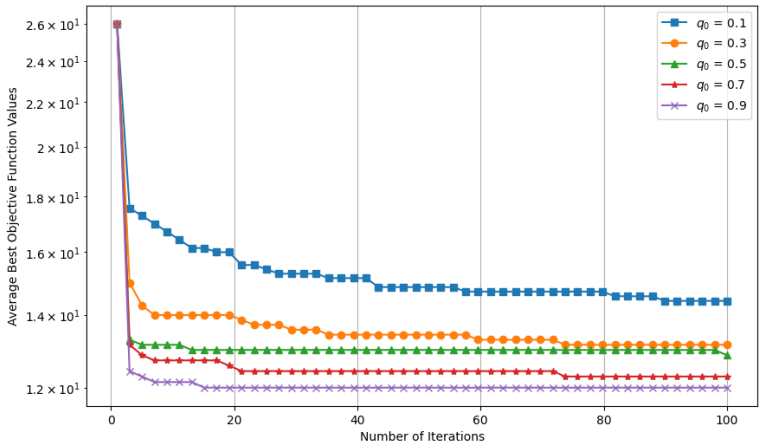

FIGURE 3. Impact of q_0

equations (3.1) and (3.2). Our experiment here focuses on how changing the threshold q_0 affects the performance of the ants and aims to determine the best value for q_0. We use 100 ants per iteration and have summarized the parameter settings in Table 4.

TABLE 4. Parameter settings for evaluating the impact of q_0

Parameters	Values
β	1
ρ	0.3
α	0.3
ϵ	0.01

We select values for q_0 from the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. A high q_0 value increases the likelihood of ants engaging in exploitation, leading to a more greedy search, while a low q_0 promotes exploration, giving ants more opportunities to discover potentially optimal solutions but risking the possibility of missing a nearby optimal solution. Figure 3 illustrates that lower q_0 values result in slower convergence to better solutions in our problem. This outcome is expected because a higher q_0 encourages exploitation, which typically leads to faster convergence, albeit potentially to a local minimum.

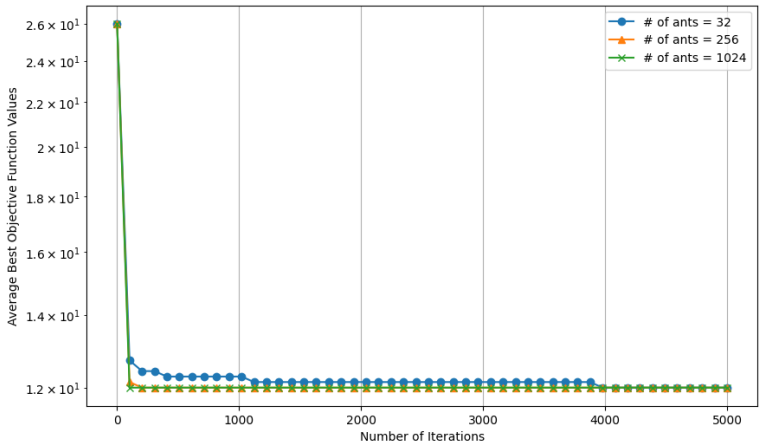

FIGURE 4. Number of ants variations: semi-log experiment chart

In our highly constrained problem, conducting extensive exploration may be less beneficial because the ants have already found an efficient path. For example, the ants may have already discovered the best services or the most effective order for selecting doctors, which leads to a higher concentration of pheromones in those areas. As a result, the ants tend to utilize these established paths rather than seeking new ones.

4.4. **Impact of the number of ants.** Research in [20] suggests that increasing the number of ants substantially enhances the algorithm's solution quality when the number of ants falls within a specific range. This experiment aims to determine whether this effect is evident in our particular problem and how it manifests. To explore this, we exponentially increase the number of ants, testing values of n in the set $\{32, 256, 1024\}$. The parameter settings for this experiment are detailed in Table 5, and the results are illustrated in Figure 4.

TABLE 5. Parameter settings in the experiment for the impact of number of ants

Parameter	Value
β	1
ρ	0.3
α	0.3
ϵ	0.01
q_0	0.7

It is observed that having more ants can lead to better solutions in fewer iterations, as they generate more solutions per iteration. In a specific area of the graph, we can see that with 1024 ants, the optimal value of 12 is achieved in approximately 30 iterations. When using 256 ants, the optimal value is reached in about 106 iterations, and with 32 ants, it takes roughly 3887 iterations to get the best result. It's important to note that increasing the number of ants from 256 to 1024 does not result in a corresponding decrease in the number of iterations required to find the optimal solution despite the fourfold increase in the number of ants. When there are only 32 ants, it takes significantly longer to achieve the optimal value. This is because pheromone values on the best routes are updated only after all ants have completed their trials. With fewer ants, there are fewer chances to find the best solution in a single iteration, which limits their ability to explore the best path after global updates and leads to a noticeable increase in search time.

5. Conclusion

We addressed a complex doctor rostering problem involving managing diverse human resources across various specialties and meeting different business demands. To solve these complexities, we utilized the ACS method for solving optimization problems. We also used exponential functions as heuristic information factors to create shift schedules that met the specified constraints. We observed that the heuristic information factor η played a crucial role in achieving optimal solutions, significantly improving the convergence rate and leading to better results. In addition, we conducted parameter tuning to determine the most effective configurations. Unlike other problems, such as the TSP, where optimal parameter settings are more apparent due to fewer constraints and more flexibility in path selection, our problem's strict constraints make enhancing solutions solely through exploration challenging. Furthermore, our proposed mathematical framework leads to the opportunity to compute optimal solutions in the future using well-known commercial optimization software packages, such as Gurobi and CPLEX.

Statements and Declarations

The authors declare that they have no conflict of interest, and the manuscript has no associated data.

Acknowledgments

The second author received support from the National Center for Theoretical Sciences of Taiwan and the National Science and Technology Council through grant 112-2628-M-006-009-MY4. The authors are thankful to Dr. Yu-Ju Lin at Taichung Veterans General Hospital for his helpful discussions about aspects of this work.

References

[1] N. Abd-Alsabour, H. Hefny, and A. Moneim. Heuristic information for ant colony optimization for the feature selection problem. In *IEEE Conference Anthology*, pages 1–5. IEEE, 2013.

[2] S. Achmad, A. Wibowo, and D. Diana. Ant colony optimization with semi random initialization for nurse rostering problem. *International Journal for Simulation and Multidisciplinary Design Optimization*, 12:31, 2021.

[3] E. Burke, P. De Causmaecker, and G. Vanden Berghe. A hybrid tabu search algorithm for the nurse rostering problem. In B. McKay, X. Yao, C. S. Newton, J.-H. Kim, and T. Furuhashi, editors, *Simulated Evolution and Learning: Second Asia-Pacific Conference on Simulated Evolution and Learning*, volume 21, pages 187–194, SEAL'98, Canberra, Australia, 1999. Springer Berlin, Heidelberg.

[4] B. Cheang, H. Li, A. Lim, and B. Rodrigues. Nurse rostering problems–a bibliographic survey. *European Journal of Operational Research*, 151(3):447–460, 2003.

[5] A. Colorni, M. Dorigo, and V. Maniezzo. A genetic algorithm to solve the timetable problem. *Politecnico di Milano, Milan, Italy TR*, pages 90–060, 1992.

[6] M. Dorigo. Positive feedback as a search strategy. *Technical Report*, pages 91–16, 1991.

[7] M. Dorigo and L. M. Gambardella. A study of some properties of ant-q. In H. M. Voigt, W. Ebeling, I. Rechenberg, and H. P. Schwefel, editors, *International Conference on Parallel Problem Solving from Nature*, pages 656–665, Heidelberg, 1996. Springer.

[8] M. Dorigo and L. M. Gambardella. Ant colonies for the travelling salesman problem. *Biosystems*, 43(2):73–81, 1997.

[9] M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[10] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[11] M. M. Flood. The traveling-salesman problem. *Operations Research*, 4(1):61–75, 1956.

[12] W. J. Gutjahr and M. S. Rauner. An ACO algorithm for a dynamic regional nurse-scheduling problem in austria. *Computers & Operations Research*, 34(3):642–666, 2007.

[13] H. Jafari and N. Salmasi. Maximizing the nurses' preferences in nurse scheduling problem: mathematical modeling and a meta-heuristic algorithm. *Journal of Industrial Engineering International*, 11:439–458, 2015.

[14] R. M. Karp. Reductibility among combinatorial problems. In E. M. Raymond, J. W. Thatcher, and J. D. Bohlinger, editors, *Complexity of computer computations*, pages 85–103. Plenum Press, New York, 1972.

[15] Z. Liu, Z. Liu, Z. Zhu, Y. Shen, and J. Dong. Simulated annealing for a multi-level nurse rostering problem in hemodialysis service. *Applied Soft Computing*, 64:148–160, 2018.

[16] M. Moz and M. V. Pato. A genetic algorithm approach to a nurse rerostering problem. *Computers & Operations Research*, 34(3):667–691, 2007.

[17] A. Oughalime, W. R. Ismail, and L. C. Yeun. A tabu search approach to the nurse scheduling problem. In *2008 International Symposium on Information Technology*, volume 1, pages 1–7, Kuala Lumpur, Malaysia, 2008. IEEE, New York.

[18] J. Puente, A. Gomez, I. Fernández, and P. Priore. Medical doctor rostering problem in a hospital emergency department by means of genetic algorithms. *Computers & Industrial Engineering*, 56(4):1232–1242, 2009.

[19] K. Shrivastava and S. Kumar. The effectiveness of parameter tuning on ant colony optimization for solving the travelling salesman problem. In *2018 8th International Conference on Communication Systems and Network Technologies (CSNT)*, pages 78–83, Bhopal, India, 2018. IEEE.

[20] R. Skinderowicz. Improving ant colony optimization efficiency for solving large tsp instances. *Applied Soft Computing*, 120:108653, 2022.

[21] T. Stützle and M. Dorigo. Aco algorithms for the traveling salesman problem. *Evolutionary Algorithms in Engineering and Computer Science*, 4:163–183, 1999.

[22] J. Thornton and A. Sattar. Nurse rostering and integer programming revisited. In *International conference on computational intelligence and multimedia applications*, pages 49–58, 1997.

[23] A. M. Turhan and B. Bilgen. A hybrid fix-and-optimize and simulated annealing approaches for nurse rostering problem. *Computers & Industrial Engineering*, 145:106531, 2020.

[24] K. Y. Wong and Komaruddin. Parameter tuning for ant colony optimization: a review. In *2008 international conference on computer and communication engineering*, pages 542–545. Institute of Electrical and Electronics Engineers, United States, 2008.

[25] J.-j. Wu, Y. Lin, Z.-h. Zhan, W.-n. Chen, Y.-b. Lin, and J.-y. Chen. An ant colony optimization approach for nurse rostering problem. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1672–1676, 2013.