Int. J. Res. Ind. Eng. Vol. 7, No. 4 (2018) 396–422

International Journal of Research in Industrial Engineering
www.riejournal.com

Development of a Doctor Scheduling System: A Constraint Satisfaction and Penalty Minimisation Scheduling Model

T. Chawasemerwa[1], I. W. Taifa [2*], D. Hartmann[1]

[1]Department of Mechanical, Industrial and Aeronautical Engineering, University of the Witwatersrand, Johannesburg, South Africa.
[2]Department of Mechanical and Industrial Engineering, College of Engineering and Technology, University of Dar es Salaam, Tanzania.

A B S T R A C T

Doctor scheduling is a complex, costly and time-consuming exercise. This study develops a constraint satisfaction and penalty minimisation scheduling model for meeting 'hard constraints' and minimises the cost of violating 'soft constraints', i.e. the user inputs, the total number of doctors to be scheduled, the maximum penalty to be met, and the minimum number of doctors to be assigned per shift. The algorithm creates a schedule which checks against all the constraints. The total schedule penalty associated with the constraint violations should be less than or equal to the user input penalty. If this condition is met, the schedule gets produced as the final and near-optimal solution. The model is managed to create a near optimal schedule with the minimal rule violations. However, it is challenging to provide a schedule with no rule violations. Such a situation is shown by the amount of computational time required to create a zero-penalty schedule, hours or even days needed to create a zero-penalty schedule. The system creates a schedule for a short period (weekly schedule) to promote flexibility; however, such a system does not promote fairness. Fairness is achieved through a cyclic schedule with rotations equal to the total number of doctors being scheduled. The system is managed to create a streamlined and flexible working environment and helped to improve the quality of healthcare. An optimization protocol can be incorporated into the system to reduce the search space and get the best optimal schedule since it is possible to get many schedules under the same user-defined parameters.

Keywords: Scheduling system, Optimisation protocol, Model development, Healthcare centres, Particle swarm optimisation, Constraint satisfaction model, Soft constraints.

Article history: *Received: 08 July 2018* *Revised: 03 September 2018* *Accepted: 23 December 2018*

* Corresponding author
E-mail address: taifaismail@yahoo.com
DOI: 10.22105/riej.2018.160257.1068

1. Introduction and Background

Constructing work schedules for high pressure working environments, such as hospital emergency rooms, is not an easy task. The scheduling in such environments needs to take into consideration a lot of conflicting rules, concerned with satisfying numerous aspects. For example, the restrictions regarding the consecutive night shifts, off days, and doctor's preferences. Efficient scheduling simplifies the manual planning. Once designing a scheduling system, then such a system should be able to satisfy users [1] and satisfaction to users can result in better performance [2]. There is no one size fits all solutions for all scheduling problems. Each problem is unique, conferring to the constraints of the particular problem (personnel policies, legal regulations, personal preferences and priorities, objectives in the scheduling, hospital policies, etc.); hence, the solutions are also unique and problem specific [3]. The task of sporadically creating a schedule commences with considering both the number of resources (workforce, usually doctors in this case) and 'a set of features to be considered in order to make use of these resources (structure of work-shift types, holidays, hard, and soft constraints to be satisfied)' [3]. Numerous approaches have been deployed to solve peoples scheduling problems. Some of the approaches include the following: Constraint logic programming [4], integer programming [5], constructive heuristics [6], genetic algorithms [7], expert systems [8], simulated annealing [9], set partitioning, and simple local search [3]. The majority of these techniques provides the near-optimal solutions concerning the conditions that are well-defined in the problem definition, despite these not necessarily being the best solutions.

1.1 Motivation

Doctor scheduling is a complex, time consuming, formidable, and daunting task. Three out of three visited hospitals create their schedules manually. Hence, the need is to introduce software that can be used to automate the scheduling process and so reduce the time and cost spent in creating the manual schedules. Manually created schedules also create room for error and bias towards certain individuals-doctors. Generally, the manual process of creating schedules can result in making some doctors overworked. Additionally, such a process can lead to some doctors working fewer hours. Again, this can also compromise the quality of the healthcare being provided to the general public. In rural hospitals, the situation is escalated by poor working conditions. Mr and PJT [10] argue that 'substantial after hour duties, an excessive workload and a perceived lack of management support impact negatively on doctor's view of working' in rural hospitals. Given the fact that currently there is a skills shortage of healthcare practitioners in South Africa, creating an optimal working environment through automated timetables using 'industrial engineering' techniques and computer-aided tools can help retain the current staff in healthcare and improve the current working conditions. This study presents a constraint satisfaction model and penalty minimisation for scheduling doctors with the intention of creating a streamlined and flexible working environment. The rules required to build the algorithm or model were derived from the literature exploring doctor scheduling systems.

1.2 Problem Statement

The scheduling of the medical professionals is a complicated matter. A hospital operates without interruption; this means that a sufficient number of medical professionals must be on duty at all hours. Such a scenario has to be balanced against a shortage of resources. As it stands some doctors work as much as 300 hours per month. Such a situation might be due to that the majority of the hospitals create schedules manually and leave a great chance for errors occurrence in making normal distributions for all available doctors. This study aims to develop a model which can help in scheduling different shifts for medical resources. In doing so, the model must be able to create consistency in distributing tasks to all doctors without any bias. Therefore, such a model should be able to balance all shifts for the available doctors.

1.3 The Significance of the Study

- To ensure that a sufficient number of doctors are on duty at all times.
- To reduce overtime and ensure that doctors recuperate from long shifts.
- To prioritise doctor preferences bearing in mind the need for the hospital to provide quality medical care to the public.
- To create a streamlined and flexible work environment.

2. Theoretical Orientation

2.1 Algorithms and Techniques Used in Doctor Scheduling

There has been much attention to the scheduling of nurses [11] and recently on the scheduling of emergency room doctors [12]. Doctor scheduling is considered as ‘one-step’ procedure where the demands from a doctor (hard constraints) must be achieved and minimising the violations to the other scheduling guidelines (soft constraints). Mathematical Programming (MP) tools are reported to be successful in scheduling doctors shifts in emergencies rooms [13]. The MP for scheduling emergency room doctors manages to include more rules than manually creating the schedule simultaneously. The emergency room doctor scheduling through MP approach can be summarised as follows [13].

- Find the rules that are not being met by the present schedule.
- Add all the necessary constraints to avoid the rule violations.
- Use the bound and branch technique in getting the new-fangled schedule, which is better than the preceding ones and mollifies more guidelines (rules).
- Repeat the iterative process until the branch and bound method cannot find any possible schedule.

It is also possible to use Constraint Programming (CP) [14] to solve numerous sophisticated scheduling problems especially if the values considered are finite [15]. In doing so, it is vital to save and update the domain of all variables when is computed for its progression. Such a step

can be performed using the specific variable constraints together with other modified variables. Filtering algorithms are used with these constraints to eliminate all values of the variables which are not consistent from the domain. Hence, all 'infeasible solutions' are removed to remain with the feasible solutions.

CP [14] was deployed successfully in creating doctor scheduling by numerous researchers. For example, Rousseau et al. [16] solved a physician rostering problem using the hybrid algorithm. Cangini [17] researched on 'a constraint programming local search algorithm for physician scheduling' and Trilling [18] developed automatic scheduling for the Cote-des-Neiges hospital in Montreal. Moreover, Bourdais et al. [19] researched a 'constraint programming application to staff scheduling in health care'. The general algorithm generated by Rousseau et al. [16] is considered dual generic constraints: Pattern constraints and distribution constraints. Rousseau et al. [16] succeeded to apply CP algorithm in a physician.

Tabu Search (TS) is also highly suitable for solving complex combinatorial problems [20]. Generally, TS is an 'iterative search procedure that starts from an initial feasible solution progressively and improves it by applying a series of local modifications' [20]. The TS was successfully used to solve physician scheduling problems. The technique was used to generate cyclically [21] and acyclic doctor schedules [22].

There are generally two distinct categories of scheduling techniques, namely 'cyclic rosters' and 'non-cyclic rosters'. In constructing cyclic rosters, the member of personnel is considered interchangeable; the personnel then rotate on the schedules until every member is assigned to each generated schedule. Cyclic rosters promote equity since all the physicians have worked the same number of shifts when the entire rotation is complete. One of the disadvantages of cyclic scheduling is that it is very inflexible, making it difficult to take outside engagements or to take doctors preferences into account [11]. For the non-cyclic or acyclic rosters, a personalised schedule is given for each person according to other personal or external engagements and preferences. This scheduling technique allows for a few personal favourites, allowances for vacation, days off requests, and working assignments during over weekend [12].

Particle Swarm Optimisation (PSO) is a nature-inspired optimisation algorithm which was also used to solve scheduling problems for an emergency room physician [23]. PSO is a population-based 'stochastic optimisation' method stimulated by the social behaviour of 'bird flocking or fish schooling'. The PSO shares similar features with evolutionary techniques, e.g. 'Genetic Algorithms (GA)' [24].

Mixed Integer Linear Programming (MILP) and constraint programming were also successfully used in the scheduling of anaesthesiology nurses [25]. The MILP was found to be superior in finding a perfect, i.e. a near optimal solution with lower computational time compared to CP.

Several existing software packages have been successful in doctor scheduling (see Table 1).

Table 1. Scheduling Software.

Software package	Source
Docs for windows for scheduling emergency room doctors.	[26]
Epsked ByteBloc medical software for generating full schedules for emergency doctors.	[27]
Tangier emergency physician scheduling software.	[28]
Clairvia physician scheduler.	[29]
Physician scheduler 4.0 by Sana-Med for scheduling emergency room doctors.	[12]

2.2. Generic Constraints in Doctor Scheduling

The constraints can be categorised as Hard (compulsory) Constraints (HC) which should be satisfied at all times and Soft Constraints (SC) which can be violated if there are conflicting scenarios to generate a workable solution. The real world situations do incorporate a large number of SC and these SC should preferably be satisfied, but can also be violated to some degree. It is a very exceptional nature to find a schedule that meets all SC [11]. The HC and SC can be further categorised into four different groups of constraints as explained by [30].

- Supply and demand constraints [30]. Constraints of this type deal with the availability of doctors. There are two generic constraints encountered in all doctor scheduling cases. The first constraint ensures that an adequate number and shift variation are run all over the planning period with the intention of guaranteeing the least coverage. The second constraint, given the doctor's level of experience, full or part-time status, and requested leave days, is not always available. Example of a Generic Demand Constraint (GDC) is as follows. During the entire planning period, ensures that precisely one doctor covers every shift. Also, GDC is regarded as HC, and many hospital departments face it. There are three variants of the mentioned scenario. Two of the main variants are as follows [12].
  - The uniformity aspect: The total number of needed workers-doctors, nurses, etc. be fixed throughout the week.
  - The non-uniform aspect: The total number of needed workers (i.e. doctors, nurses, etc.) be fixed from Monday to Friday.

  Example of a generic availability constraint is as follows. For the entire planning horizon, all the preferences of every doctor should be met. There are four types of doctor preferences: Pre-assignments, vacations, forbidden assignments, and preferences or aversions.
- Workload Constraints (WC). Constraints of this type are meant to cater for the days or shifts that are assigned to the doctors in a defined planning horizon. For example, the number of shifts each doctor attends to per month. WC include the number of shifts or the number of hours that are assigned to the doctors within a week, month, or the entire planning horizon. The following are examples of WC:

- Limits on WC: Throughout the given planning horizon, a doctor should be assigned a line of work that lies within a specific interval. Examples of this include a doctor who is assigned to work 28 hours in a week can accept to work up to 32 hours in a week and at most four shifts are assigned to a doctor in any given week. The limits on WC are either there because of terms signed within a contract, or to encourage a consistent workload. These are considered to be SC. This constraint also encourages uniform workloads.

- Fairness Constraints (FC). Constraints of this nature ensure that the workload such as the number of night shifts assigned to doctors with the same level of experience, is the same for a given planning period. The FC ensures the fair allocation of shift types among doctors of the same type and/or experience. Examples of FC include 'distribution of type of shifts constraints'. For an entire horizon, the same kind of planned shifts-weekend or night shifts must be equitably allocated with consideration to experience level.
- Ergonomic Constraints (ERC). Always, it is good to consider ergonomic factors [31,32]. The ERC is a set of rules that ensure that schedules with defined quality standards are produced. The ERC constitutes a large number of constraints. These rules allow a certain level of quality for the generated schedules and maybe global constraints or targeted at the specific individuals [33,34].

2.3 Basic Conditions of Employment Act

The underlying 'conditions of employment act' applies to all employers and employees. It regulates the leave days, working hours, deductions, employment contracts, pay slips, and termination [35]. According to reference [36], the following general working conditions should be satisfied in the workplace. An employer should not allow an employee to work more than 45 hours within a working week and 9 hours in any working day given that the employee works 5 days or less in a week. In the event that the employee works more than 5 days a week, he or she should not be allowed to work more than 45 hours in a week and 8 hours a day. Certain agreements can be made between an employer and the employee to extend the daily working hours by up to 15 minutes in any day but to less than or equal to 60 minutes a week. Such a situation allows the employee to continue with other non-work-related duties which include serving the public which is performed after working hours. Generally, this leads to the reduction of 8 ordinary working hours in a day and 40 working hours per week.

With regards to overtime, an employee should not work more than 3 hours as overtime in any given day, or to have accumulated more than 10 hours of overtime per week unless there is a special signed agreement. Given that an employee has worked overtime, he or she should be paid no less than 1.5 times the general wage for the overtime worked for. In the event of a compressed working week, some written agreements may require employees in this case doctors to work up to 12 hours for one day. Generally, there are so many regulations regarding the 'conditions of employment act' in South Africa; therefore, the authors did not incorporate all the rules as required. However, more information can be found in the given reference [36].

2.4 Software Development Process Models (SDPM)

SDPM is an 'abstract representation' of the actual process. The SDPM presents a process portrayal from a specific perspective such as specification, design, validation, and evolution. Various SDPM exist including the Waterfall Model (WM), the iteration model, V-shaped model, spiral model, and extreme model [37]. The WM shown in Figure 1 was deployed as the software development model for this study [37].

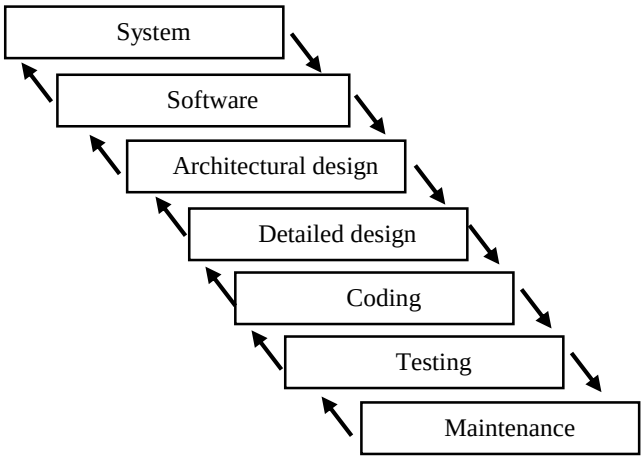


Figure 1. The WM Model [37].

The WM encourages planning in early stages, and hedges against the design flaws before they are developed. WM was introduced by Royce in 1970 [38]. The WM lifecycle consists of several non-overlapping stages. WM begins with establishing the system requirements and software requirements, followed by software requirements, architectural design, detailed design, coding, testing, and maintenance. Table 2 shows WM stages [37].

The waterfall model has got its advantages and disadvantages as a software development process. Advantages: (a) Can be easily understood, (b) It is extensively deployed and known in theory, and (c) It strengthens decent habits such as for defining before design and design before coding, and (d) It identifies deliverables and milestones [38]. Disadvantages: (a) It is too idealised and it does not match well with reality, (b) WM does not reflect the exploratory nature of development, (c) Impractical to expect accurate necessities very early in the project, (d) It is challenging to integrate risk management, (e) It delays the discovery of serious errors, and (f) It is hard and costly to make variations to the used documents [38, 39].

Table 2. The Waterfall Model [37].

#	Stage	Description
A	System requirements	Establishes all necessities that are needed in creating or developing a particular model. This can include software tools, hardware requirements, and so forth.
B	Software requirements	Helps in establishing all necessities especially for the aspect of software functionality. It is important to think about compatibility and technical limitations which can happen when synchronising from other databases and applications, user interface requirements as well as the general performance of the model to be developed.
C	Architectural design	Helps in coming up with the 'software framework' with the intention of having a well-defined significant components. Also, a designer must consider any aspect related to external interfaces together with other tools required in executing such a project.
D	Detailed design	Performing accurate observations on each 'software component' defined in the 'architectural design' phase.
E	Coding	Facilitate implementation of the detailed design specification by creating software through a chosen programming language.
F	Testing	It is used to determine whether the software has achieved the required specifications.
G	Maintenance	Upkeep process whenever there are observed or detected problems in the developed software.

3. Model Development

3.1 System Specifications

Requirements

- The system should be able to schedule a user-specified number of doctors.
- Create a schedule for a week's planning period.
- The system should be able to create a schedule with a total rule violation penalty less than or equal to a user set penalty.
- The system should meet the minimum set number of doctors per shift. In practice, a minimum of 1 or 2 doctors should cover every shift.
- Meet doctor's preferences for off days.
- Avoid the rule violations which are not permitted by the law, e.g. avoid assigning consecutive evening and night shifts.
- All shifts should be evenly distributed across the entire planning period.

Constraints include time, scope and cost

Criteria for the model to be developed are as follows

- The system should be user-friendly (user-friendly interface) and easy to use.
- Low system computational time.
- Developing a model which creates a high-quality schedule with little rule violations.
- Developing a low-cost model which will use low maintenance cost.
- It should be easy to make any rule changes or add more rules to the system, i.e. the system should be flexible.

3.2 Problem Description

A Constraint Satisfaction model and Penalty Minimisation Model (CSPMM) is needed. An objective function required to be formulated and constraints should be derived from the literature. A day is comprised of three shifts, namely the 'day', 'evening', and the 'night' shift. The shift times are defined as follows: Day shift from 08:00 to 16:30; evening shift from 16:00 to 00:30; and night shift from 00:00 till 08:30. Set of values (notations) for the deployed input data are as shown in Table 3.

3.3 Objective Function

The objective of the system is to find a near optimal weekly doctor schedule. All the hard constraints should be met if possible and the cost of violating the soft constraints should be minimised. The doctor`s preferences are turned into constraints and the preferences are assigned the same weight regardless of the doctor's level of experience. Both hard constraints and soft constraints are assigned violation penalties or costs. Maximum or higher value penalties or costs are assigned to violating hard constraints and lesser penalties or costs are assigned to violating the soft constraints. Eqs. (1) and (2) show the objective function for minimising the total violations to both hard and soft constraints.

$$F(X) = \sum_{1}^{d} \sum_{1}^{s} \sum_{1}^{d_y} C_{dsd_y} \tag{1}$$

The following criteria should be met.

$$(F(X) = \sum_{1}^{d} \sum_{1}^{s} \sum_{1}^{d_y} C_{dsd_y}) \leq C_s \tag{2}$$

The above formulation was used in the subsequent sections which define the hard and soft constraints of the model and the respective penalties associated with violating all hard constraints as well as some of the soft constraints.

Table 3. Description of the Used Set of Values and Parameters.

Notation	Description
S	Shifts of the day ($S_1, S_2, \ldots, S_n$).
C	Penalty or cost of violating a constraint.
C_s	User set the maximum penalty.
d	Doctors.
O	Supply of doctors.
P	Doctor`s preferences.
d_y	Days of the planning horizon.
S_1	Specific set S_1.
S_n	Specific set S_n.
X_{dsd_y}	Assignment of doctor d, shift s of the day d_y.
$X_{d\ldots sd_y}$	$\{$If Doctor d works on shift s on day d_y (with $S_1 . S_2 . \ldots S_n$) and $X_{d\ldots sd_y} = 1$; Otherwise $= 0$.
P_{dsd_y}	Preference of doctor d. shift s of the day d_y(with $S_1 . S_2 . \ldots . . S_n$).
O_{sd_y}	Supply for shift s of the day d_y(with $S_1 . S_2 . \ldots . S_n$). $O_{sd_y} \geq 1$ doctor.
C_{dsd_y}	Cost or penalty for doctor d. shift s of the day d_y for violating constraints.
M_{sd_y}	Demand for shift s of the day d_y (with $S_1 . \ldots . S_n$). $M_{sd_y} \leq O_{sd_y} \geq 1$ doctor.

3.4 Hard and Soft Constraints

Hard constraints

Hard constraints are stringent requirements stipulated by the law, and these should be satisfied all the time and at all costs. The following hard constraints and their associated penalties were used to build the system, see [Eq. (3)].

- The working hours of a doctor cannot exceed 8 hours in every working day, i.e. a doctor is only allowed to work 1 shift in a day. If a doctor can work for more than 8 hours in a day, multiple shifts, then the doctor cannot have sufficient rest which is undesirable and potentially dangerous.

$$\sum_{1}^{s} X_{d\ldots sd_y} \leq 1 \; \forall \; d. \ldots . d_y \tag{3}$$

- The minimum demand of doctors in every shift of every day must be met. For example, there should be a minimum of 1 doctor on duty per shift and a maximum which is dependent on the shifting demand or daily demand of doctors. This ensures that every shift has a doctor assigned to it and hence the quality of the provided healthcare is not jeopardised. Eqs. (4) and (5) show the demand of doctors per shift. Constraint violation penalty is 1000 South African Rand (ZAR).

$$\sum_{1}^{d} X_{d\ldots sd_y} = O_{s\ldots d_y} \qquad \forall \; d. S. \ldots S_1. \ldots . S_n \tag{4}$$

$$O_{s\ldots d_y} \geq M_{s\ldots d_y} \quad \forall \; s. d_y. \ldots S_1. \ldots . S_n \tag{5}$$

Where $S_1. \ldots . S_n$ represents other peripheral requirements.

- Consecutive evening and night shifts are not allowed. As the evening shift belongs to the previous day and the night shift belongs to the new day, a constraint is needed to prevent a doctor from working evening-night shifts on consecutive days. Such shifts are not permitted by law. Eq. (6) shows consecutive day and night shifts. Constraint violation penalty is 1000 South African Rand (ZAR).

$$X_{d\ldots 2.d_y+1} + X_{d\ldots 3.d_y} \leq 1 \quad \forall \; d. d_y. \ldots S. \ldots . S_n \tag{6}$$

Whereby S =2 represents the evening shift and S = 3 represents the night shift.

Soft constraints (SC)

The following SC are considered. Some soft constraints have an associated penalty for violating the constraint. However, it is impossible to meet every soft constraint because some constraints are conflicting. Hence, some soft constraints have no associated penalty violations.

Fist SC: The total working hours of a doctor cannot exceed 48 hours (6 days) in a week and the total working hours of a doctor cannot be less than 16 hours (2 days) in a week.

Given a maximum of *y* working days and a minimum of *z* working days, within a period of any d_y working days it follows that: Eqs. (7) and (8) show the minimum and maximum working days, respectively. Constraint violation penalty is 1000 ZAR to either case.

$$z \leq \sum_{1}^{s} \sum_{1}^{d_y} X_{d\ldots sd_y} \quad \forall\, d. \ldots \tag{7}$$

$$\sum_{1}^{s} \sum_{1}^{d_y} X_{d\ldots sd_y} \leq y \quad \forall\, d. \ldots \tag{8}$$

The constraints ensure that a doctor gets enough rest days which is highly recommended for the doctor's well-being. However, the constraint alone is insufficient to enforce proper amounts of rest since it only gives the number of rest days. The rest days are in the range of 1 to 5 days in a working week. The schedule might arrange work in such a manner that the doctor works continuously within the period of d_y days. To prevent such a scenario from happening, an additional constraint is required to limit the number of consecutive working days. This constraint is known as the 'maximum consecutive work-days' and is used together with the above constraint for enforcing the rest days.

Second SC: Consecutive working days of a doctor cannot exceed 6 days, and there should be at least 2 consecutive working days before a doctor takes a day off.

Given *m* consecutive working days within the d_y day's period, there must be at least 1 rest day. Eq. (9) shows the consecutive working days.

$$\sum_{1}^{s} \sum_{m}^{m+4} X_{d\ldots sd_y} \quad < m \quad \forall\, d. \ldots \text{where } m \leq d_y - m \tag{9}$$

The constraint for Eq. (9) is further developed if the number of consecutive working days consists of night shifts. There must be at least 3 days off after a sequence of 3-night shifts. Usually, after *j* consecutive night shifts, there must be at least 1 sleep a day and *v* rest day (equivalent to *1+v* rest days, but only *v* rest days will be counted towards legally required rest days).

Eq. (10) shows the constraint on rest days after working consecutive night shifts.

$$\sum_{t}^{t+j-1} X_{d\ldots.3.d_y} + \sum_{t+1}^{t+v+j} \sum_{1}^{s} X_{d\ldots sd_y} \leq t \quad \forall\, d. \ldots \text{where } t \leq d_y - j - v \tag{10}$$

Where *s* = 3 represents the night shift.

The above constraint manifests itself in two separate forms. The first form ensures the start of the mandatory rest day for the consecutive night shifts and the second form ensures the maximum number of the continuous night shifts. For the maximum number of continuous night shifts, the number of successive night shifts should be reduced to 2 or 3. Eq. (11) shows a maximum number of the consecutive night shifts.

$$\sum_{1}^{j} X_{d\ldots3.d_y} \leq j - u \quad \forall d.\, d_y \ldots S_1 \ldots S_n \tag{11}$$

Whereby $j - u$= the number of night shifts which is considered acceptable and meets the doctor's preferences.

Third SC: Meet the doctor's preferences for off days as much as possible.

Fourth SC: All shifts must be equally shared for the entire schedule, similar to night shifts.

Fifth SC: A number of successive evening shifts should be reduced to a maximum of 4.

Sixth SC*:* There should be no more than 5 days which should be considered as consecutive off days of a doctor.

Seventh SC: Avoid arranging a shift pattern, off day-work day-off day.

Eight SC: Day shifts followed by evening shifts followed by night shifts should be promoted (forward rotation principle), i.e. a doctor should have 24 hours of rest before commencing the next shift.

Ninth SC: There must be as a minimum 2 and as a maximum 4 consecutive matching shifts.

3.5 Assumptions

- The system does not account for doctors with different levels of experience; all doctors are assumed to have the same level of experience.
- Overtime is not modelled in this system.
- A uniform case is considered, i.e. weekdays (Monday to Friday) are scheduled the same as the weekends (Saturday and Sunday) regarding the scheduling rules such as the number and the length of shifts and the minimum number of doctors per each shift.
- All staff members (doctors) are treated as permanent staff, i.e. there are no part-time doctors.
- Local holidays, regional holidays and national holidays, New Year's Eve, and Christmas Eve are all categorised as off days. The system does cater for holidays since holidays were turned into constraints as off days.
- The second consecutive off day is treated as a standby to cover for unplanned absences such as illness or the death of a close family member.
- Doctor demand is assumed to be random during any shift since it is not easy to predict the flow distribution of patients. However, it is assumed that there should be at least one doctor on duty for every shift.
- Since the rules derived from the literature are mainly for scheduling emergency departments; the system is highly suitable for use in emergency departments.

3.6 Method of Operating the System

Figure 2 shows the system flowchart. Run the scheduling model in the following order:

- Enter the number of doctors (e.g. 20 doctors) to be scheduled.
- Enter the required minimum number of doctors per shift (e.g. 0 per shift).
- Enter the maximum allowable penalty (e.g. 600 ZAR).
- Click 'get timetable' to run the scheduling system.
- Wait for the system to run until a near optimal schedule has been found.
- In the event of a near optimal solution, click 'report' to get the report of the distribution of shifts for the entire planning period.
- In the event of an infeasible solution click 'esc' key and end the programme.
- Save before closing the programme, but the code saves automatically.

In order to succeed in using the developed system, there are precautions to be observed.

- Run the programme in the outlined order.
- One should save before closing the program, but the code saves automatically.
- The system can work with a maximum of 20 doctors (20 doctors or less).
- Automatically deleting the pie chart is somewhat cumbersome; one must delete it manually before generating another pie chart.

Different scenarios were run to compute the computational times. A machine with the following specifications was used.

- Rating: 3.9 Windows Experience Index.
- Processor: Intel® Xeon® CPU X3370 @ 3.00GHz 3.00GHz.
- Installed memory (RAM): 4.00 GB.
- System type: 64-bit operating system (windows 7 professional).

4. Testing the Developed Model

In order to test or validate the developed model, a single and random scenario was run and the results were thoroughly analysed to determine whether the system is meeting the requirements (both hard and soft constraints) or not. The testing results have presented in Table 4. The following user inputs were used to generate a weekly schedule: A number of doctors (20), minimum doctors per shift (1), maximum set penalty (300), and minimum achieved penalty (200). For the weekly timetable, Table 4 was created under the parameters mentioned above.

Figure 3 was obtained using the information given in Table 4. Figure 3 shows the percentage distribution of shifts for all doctors for a weekly planning period. Figures 3 to 8 were generated using the Microsoft® Excel version 2016 while the Minitab® version 18 generated Figure 9.

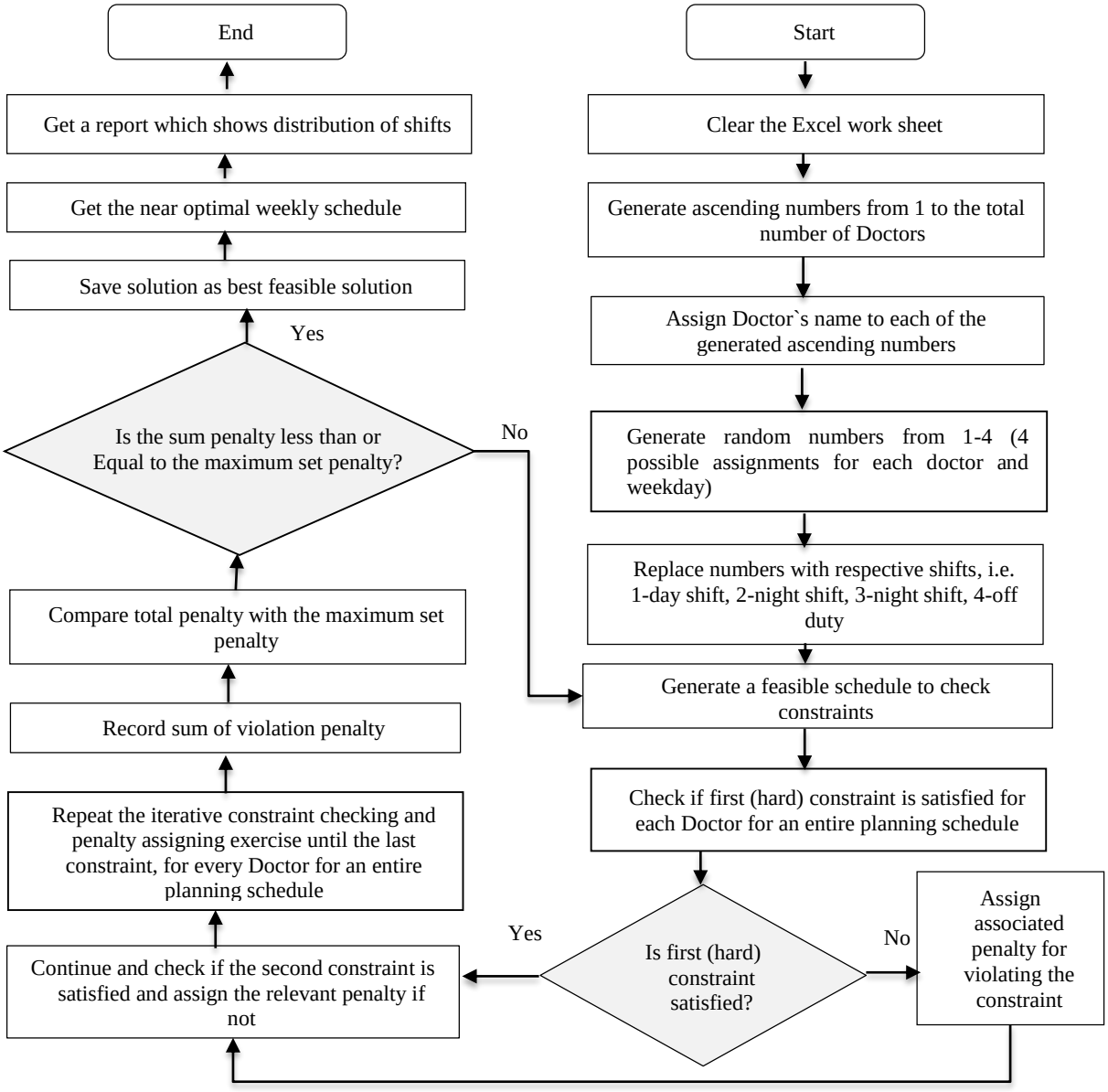


Figure 2. Methodological System Flowchart.

Table 4. Weekly Timetable.

Doctor`s names	Day Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
A	Off	Night	Night	Night	Evening	Off	Evening
B	Off	Off	Day	Off	Off	Day	Evening
C	Day	Off	Night	Day	Night	Evening	Off
D	Off	Day	Off	Off	Evening*	Night*	Day
E	Off	Day	Off	Night	Off	Night	Evening
F	Day	Night	Off	Evening	Day	Off	Evening
G	Evening	Off	Evening	Day	Evening	Day	Evening
H	Night	Off	Day	Off	Day	Off	Off
I	Day	Night	Night	Evening	Off	Night	Off
J	Off	Evening	Off	Off	Evening	Day	Night
K	Off	Off	Off	Evening	Off	Off	Evening
L	Evening	Day	Night	Off	Night	Evening	Day
M	Day	Evening	Day	Evening	Evening	Evening	Off
N	Evening	Off	Night	Off	Day	Day	Day
O	Off	Off	Evening	Day	Day	Day	Off
P	Day	Night	Off	Evening	Off	Day	Off
Q	Off	Night	Off	Day	Night	Evening	Day
R	Evening	Day	Day	Off	Day	Off	Day
S	Evening	Day	Night	Night	Night	Evening	Off
T	Off	Evening	Day	Evening	Evening	Off	Evening

Note: * Indicates the violated rule (see section 6.1). Note: A to T are the doctors` names.

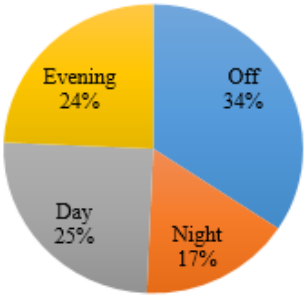
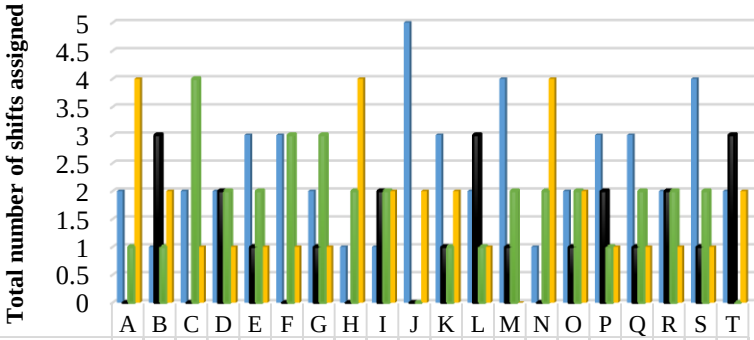


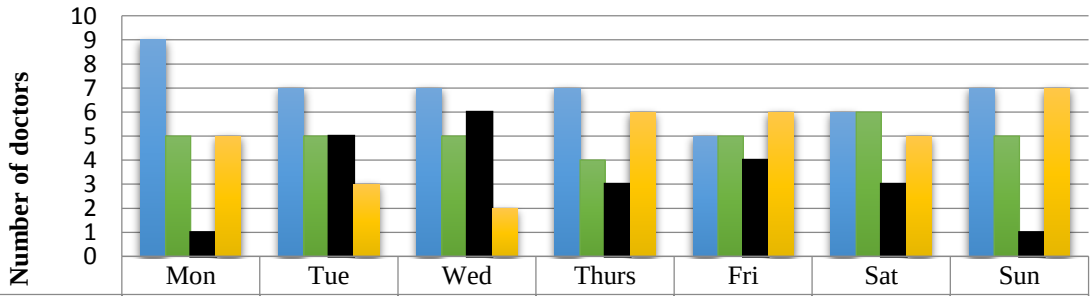
Figure 3. Percentage Distribution of Shifts.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Off shifts distribution per doctor	2	1	2	2	3	3	2	1	1	5	3	2	4	1	2	3	3	2	4	2
Night shifts distributions per doctor	0	3	0	2	1	0	1	0	2	0	1	3	1	0	1	2	1	2	1	3
Day shifts distribution per doctor	1	1	4	2	2	3	3	2	2	0	1	1	2	2	2	1	2	2	2	0
Evening shifts distribution per doctor	4	2	1	1	1	1	1	4	2	2	2	1	0	4	2	1	1	1	1	2

Figure 4. Distribution of Shifts for One Week.

[Note: A to T represent doctors` names]



	Mon	Tue	Wed	Thurs	Fri	Sat	Sun
Off	9	7	7	7	5	6	7
Day	5	5	5	4	5	6	5
Night	1	5	6	3	4	3	1
Evening	5	3	2	6	6	5	7

Figure 5. Weekly Assignments of Shifts Per Day of the Week.

Figure 4 was created using the information presented in Table 4 that shows the off, night, day and evening shifts for each doctor for a weekly planning horizon. The off days range from 1 to 5 days in a weekly planning period. Figure 5 shows the distribution of shifts per day of the planning horizon. For example, on Monday it can be seen that 9 doctors are off duty, 5 doctors are assigned to the day shift, 1 doctor is assigned to the night shift, and 5 doctors are assigned to the evening shift.

The authors have computed the total shifts each doctor works during the entire planning period. Figure 6 shows the total number of shifts regardless of type that each doctor works in a weekly planning period. It can be seen that doctors do not work the same number of shifts. In order to ensure fairness, a cyclic roster with rotation can be used to create balance. A cyclic sample schedule with rotation for 20-weeks was performed.

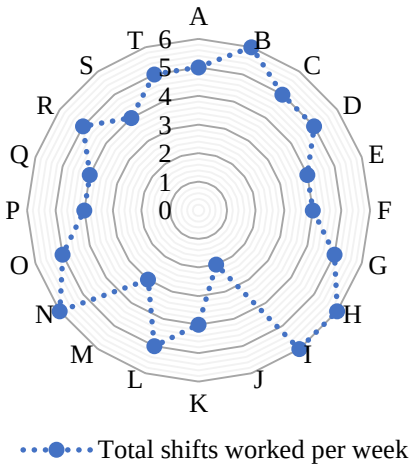


Figure 6. Total Shifts Worked Per Week.
[Note: A to T represent doctors` names]

Having shown all the necessary results, the authors then analysed the consecutive shifts per doctor for a weekly planning period (see Figure 7). The consecutive shifts regardless of type range from 0 to 3. For example, Dr 'O' was found to have 2 consecutive off shifts, 3 consecutive day shifts, and no consecutive evening and night shifts.

5. Scheduling System Analysis

Figure 8 shows the effect of varying the maximum allowable penalty on the system computational time. Three different scenarios were investigated and they comprised of the following user-defined input parameters.

- Scenario 1: Fixed system parameters, i.e. 20 doctors and a minimum of 2 doctors per shift.
- Scenario 2: Fixed system parameters, i.e. 15 doctors and a minimum of 2 doctors per shift.
- Scenario 3: 10 doctors and a minimum 1 doctor per shift.

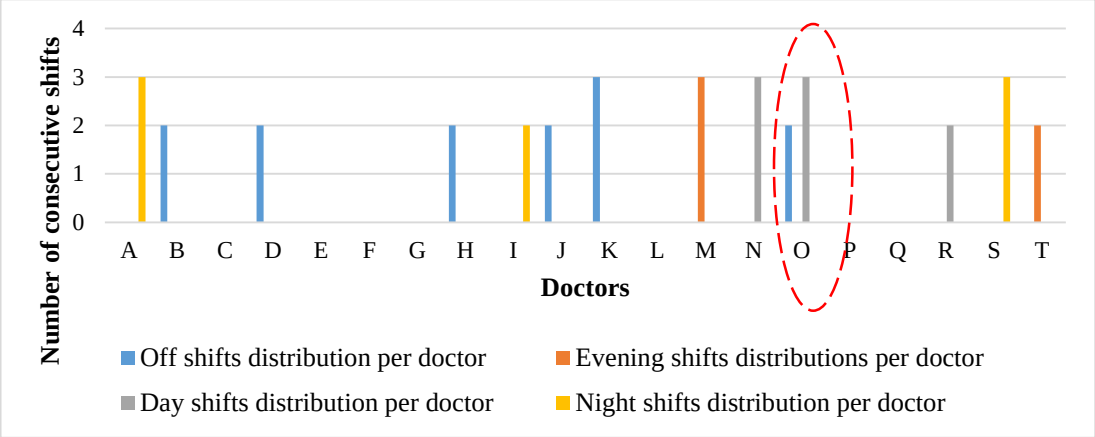


Figure 7. A Maximum Number of Successive Shifts.
[Note: A to T are the doctors` names]

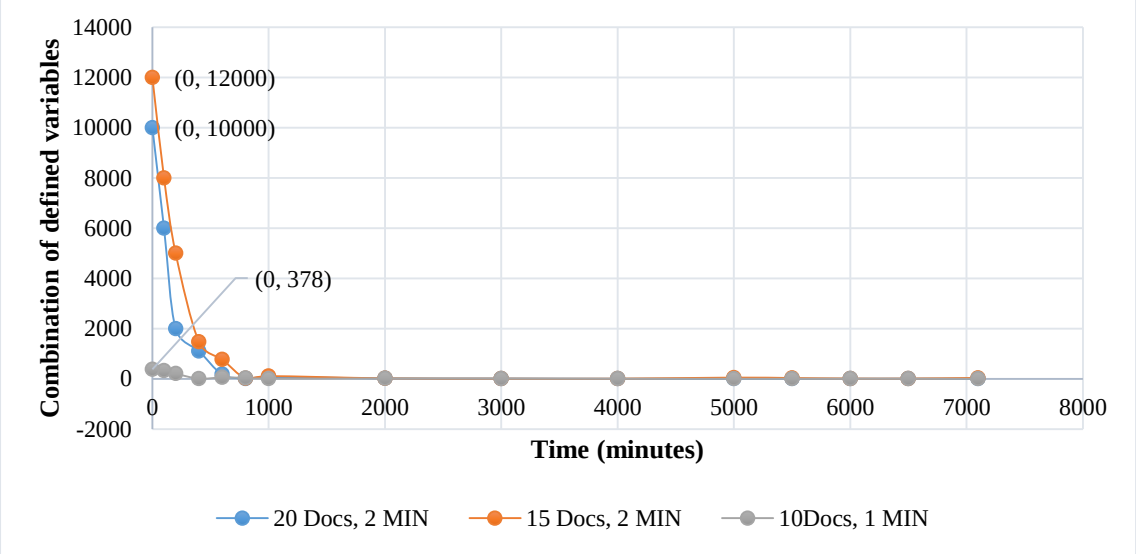


Figure 8. Computational Time (s) Plotted Against Penalties.

Figure 9 shows the normal distribution for the scheduled shifts (week 1 to week 3). The authors presented results using Minitab® version 18 to indicate how the scheduling of the four shifts follows a normal distribution curve. From Figure 9, it can be observed that for week 1 the skewness of the data indicates that the generated shifts are normally distributed for evening shifts than other shifts. However, for week 2, off and day shifts indicate that the created shifts are normally distributed. This is due to that, the majority of the days are located on the low or high side of the shown normal distribution graph (see Figure 9). In general, for all the three days, the skewness specifies that the generated schedules are normally distributed. In particular, from Figure 9, it cannot directly be concluded that the generated shifts follow normal distributions,

however, for more extended periods, data can be fitted well and adhere to normal distribution requirements. Also, Figure 9 details the mean and standard deviation for each week. However, it should be noted that other results for 17 weeks were not displayed in Figure 9.

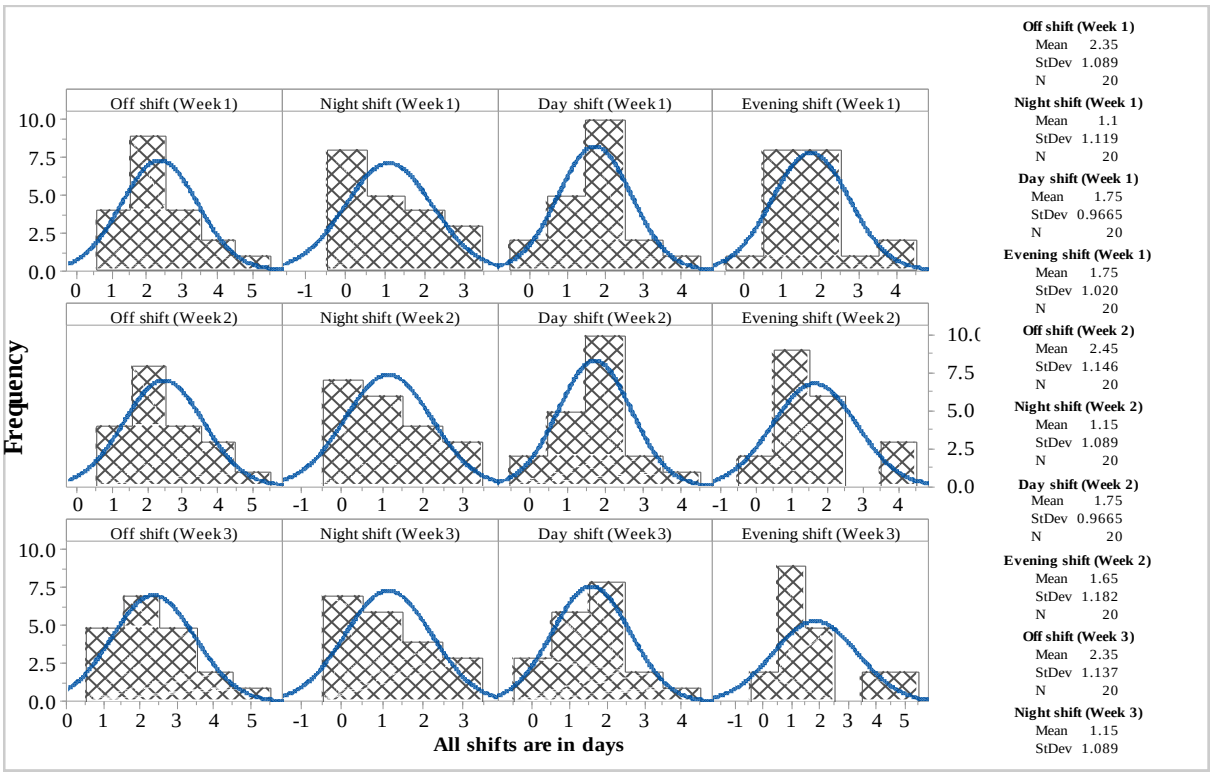


Figure 9. Normal Distribution for the Scheduled Shifts (week 1 to week 3).

6. Discussion

6.1 Testing Results Analysis

Hard constraint one (1) states that a doctor is only allowed to work a single shift in a day. It is apparent from Table 4 that the system ensures that no doctor works more than one shift in a day. Working more than one shift in a day cannot allow a doctor to have enough rest and it is highly undesirable and potentially dangerous. Although the system meets the requirement that no doctor works more than one shift in a single day; it is possible to assign doctors to work consecutive evening and night shifts which are illegal according to the law. This is possible because the evening shift belongs to the previous day and the night shift belongs to the following day. The system is constrained from arranging shift patterns as evening shifts followed by night shifts although in some instances the constraint is violated to find a scheduling solution. For example, a situation where the rule was violated is highlighted in Table 4 (see Dr 'D' for the scheduled shifts on Friday and Saturday).

Conferring to the second hard constraint, the minimum demand for doctors in every shift should be met. The user inputs the minimum number of doctors that should be scheduled by the system. The minimum input number of doctors per shift should relate to the total number of doctors to be scheduled; for example, when scheduling 20 doctors inserting a minimum of 10 doctors per shift will result in an infeasible solution. The system assumes a random demand for doctors per shift since it is not easy to model the patient arrival process per shift of the planning horizon. As shown in Figure 5, there are 5 doctors on evening shift for Monday, 3 doctors on evening shift for Tuesday, 2 doctors on evening shift for Wednesday, 6 doctors on evening shift for Thursday, etc. The number of doctors is not fixed to 1 or 2 as the literature suggests. It is also shown that there is at least one doctor assigned to every shift, which ensures that the quality of healthcare provided to the general public is not being compromised.

According to the first CS, a doctor should have at least a single day off in a weekly planning horizon, and a doctor should also work for at least two days in a weekly planning horizon. As shown in Figure 4 specifically for the off days for each doctor for a weekly planning horizon, all the doctors have at least a single day off in a weekly planning horizon allowing them to have enough rest. The maximum number of off days is 5, which means that a doctor have worked for 2 days in a weekly planning horizon. Figure 6 shows that, each doctor works for at least 2 shifts (2 days) per week and a doctor works for a maximum of 6 shifts (6 days) per week; hence meets the requirement that every doctor should have a single off day in a weekly planning horizon.

According to the second CS, the consecutive working days should not exceed 6 days in a weekly planning horizon and there should be at least 2 consecutive working days before a doctor takes a day off. This also follows that 'off day-work day-off day' patterns should be avoided. Figure 7 shows that the maximum number of consecutive working days before a doctor takes a day off is 3 consecutive working shifts regardless of the type of shift. However, there are some doctors with zero consecutive working shifts before taking a day off which shows that the off day- a work day- off day pattern is occurring in the system and this is undesirable. There should be at least 2 consecutive shifts for each doctor for the rule to be met. Since this is a soft constraint, it can be violated to some degree by the system. It is also recommended that the maximum number of consecutive night shifts be 3 and a doctor should have a day off after a series of three consecutive night shifts. From Figure 7, it can be seen that the maximum number of consecutive night shifts is also 3. However, days off are not given in some instances after a doctor has worked 3 successive night shifts.

According to the third CS, doctor's preferences for off days should be met. Doctor's preferences were turned into constraints, i.e. a doctor should have at least a day off in a weekly planning horizon and a doctor should have a day off after working 3 consecutive night shifts, etc. Since preferences for off days are random and not so easy to predict when a particular doctor requires a shift off, for example, due to the loss of a close family member, the system does not account for such. The internal arrangements amongst doctors can be sorted to accommodate such cases.

Fourth CS states that the number of successive evening shifts should be reduced to a maximum 4. Figure 7 shows that the maximum number of consecutive evening shifts is 3, hence the system is satisfying this rule.

Fifth CS states that all shifts should be equally distributed over the schedule for the entire planning period. From Figure 3, the shift distributions are as follows: 24 % evening shifts, 25 %-day shifts, 17 %night shifts, and 34 % off day shifts. It is apparent that the doctors have enough rest (off) days and the other shifts are nearly evenly distributed over the schedule.

Sixth CS states that consecutive off days for a doctor cannot exceed 5 days. Figure 7 shows that there is a maximum of 3 consecutive off days, hence the constraint is satisfied.

Ninth CS states that there must be at least 2 and at most 4 consecutive matching or same shifts. Figure 7 shows that sometimes there are no successive shifts. This indicates that the constraint is sometimes being violated. However, it is a soft constraint; hence rule violations are permissible.

6.2 General Discussion on the Developed Model

Manually creating schedules is an arduous and time-consuming exercise which can take days or even weeks. Henceforth, low system computational time is of great importance. As shown in Figure 8, CSPMM gives result in acceptable computational times when the penalty is 300 or higher. In instances where a Zero-Penalty (ZP) schedule is required, the computational system time dramatically rises to hours or even days. In many cases, it is impossible to find a feasible solution under ZP conditions. A schedule obtained under ZP conditions is nearly optimal, and all the system constraints could have been satisfied by the system, but this is highly unlikely because some constraints are conflicting and there are many decision variables. As the penalties go beyond 1500, it implies that the system can override many system constraints; hence this results in a very low system computational times and might also result in some hard constraints not being met and the production of a schedule that is far from being optimal.

Consider the minimum number of doctors per shift corresponds to the number of doctors to be scheduled. For example, when scheduling 20 doctors, if the user proposes that the minimum number of doctors per shift should be 10, the system cannot find a possible schedule. Ten doctors per shift imply that the number of doctors to be assigned per day is 30. The total number of doctors to be assigned per day should be less than the total number of doctors to be scheduled for the system to find a feasible solution. The user only has to define the minimum number of doctors to be assigned per shift and the system assigns the actual number of doctors per specific shift. This avoids fixing the number of doctors to be assigned per shift to be 1 or 2 as the literature suggests. Assigning more doctors to shifts yet meeting all the other constraints, for example, for off days such an instance can help to improve the quality of health care being provided by hospitals. Also, this can reduce the workload to individual doctors and help to reduce working overtime work. As shown in Table 4, the system permits the rule violations to some degree in order to find near-optimal schedules. As shown in some cases the evening-night shifts are

permissible. Eliminating the rule violations (zero penalty schedules) and fixing the system, results in extensive computational times and the system will not be able to find a near optimal schedule. In summary, the CSPMM generated is illustrated in Figure 10 and the same figure shows the key contributions for this study.

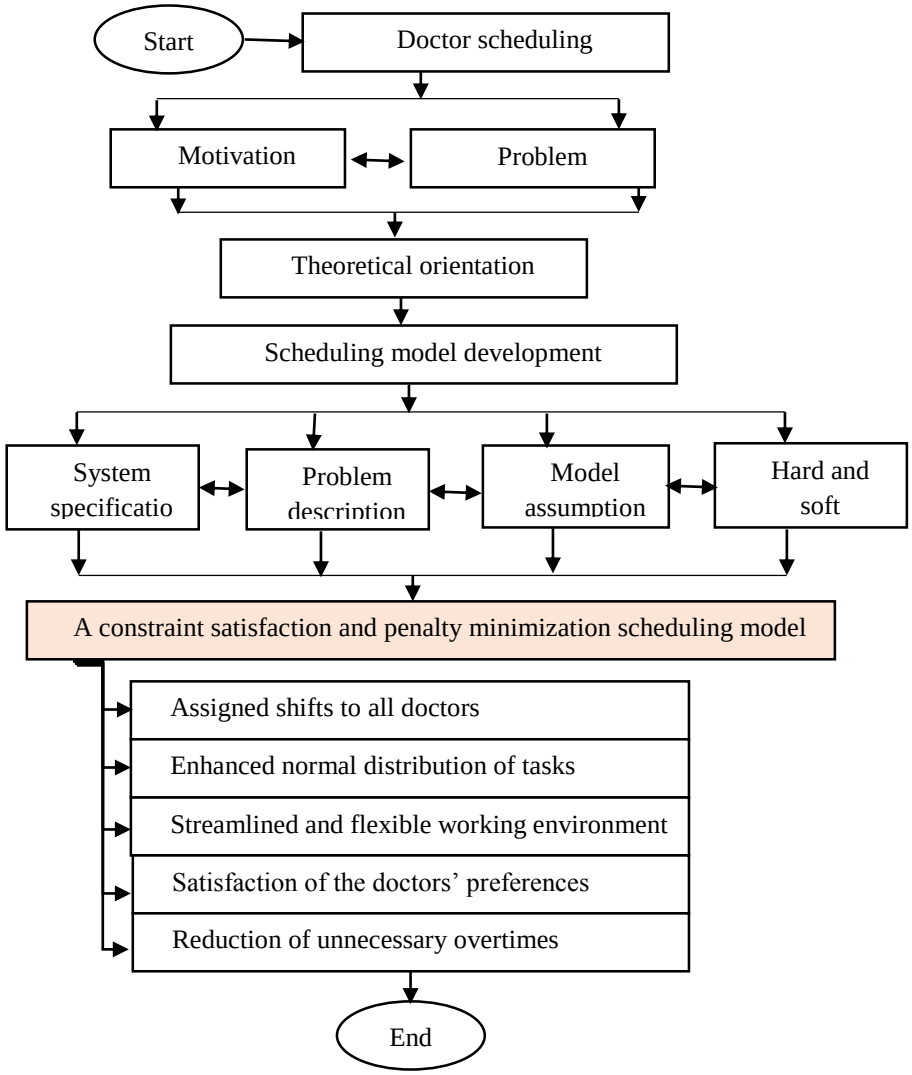


Figure 10. Enhanced Contributions through the CSPMM.

7. Conclusion

7.1 Concluding Remarks

The CSPMM provides the near-optimal schedules effectively with reasonable computational times compared to manually creating schedules. In order to develop the model, some simplifying assumptions were made. Assumptions are necessary in order to simplify the problem and develop

a manageable problem scope. However, assumptions should be kept at a minimum so that the developed solution was close to the actual solution. The assumptions may affect the model to generate a schedule that is far from being realistic. The CSPMM created a weekly schedule, as shown in Table 4. A schedule created for a shorter period allowed for flexibility. For example, in cases where a doctor was absent for more than 7 days, he or she can be eliminated from the schedule. In order to promote fairness, a cyclic schedule with rotation can be created from the weekly schedule. One main advantage of a cyclic schedule with rotation was the fact that after a complete cycle all doctors would have worked the same number of night shifts, day shifts, evening shifts, and had the same number of off days. The disadvantage of such a system is the lack of flexibility, but it is 100% fair in distributing shifts.

7.2 Practical Implications and Research Contributions

The study discussed CSPMM. Such a model was very useful for scheduling doctors in healthcare centres and hospitals. Hospitals which deal with the scheduling of three shift patterns, namely the ‘day shift’, ‘evening shift’, and the ‘night shift’ can benefit from such a model. The shift times were defined as follows: Day shift from 08:00 to 16:30; evening shift from 16:00 to 00:30; and night shift from 00:00 till 08:30. These shift times can be altered depending on the need. The developed system managed to create a streamlined and flexible working environment and helped to improve the quality of healthcare being provided to the public by ensuring that at least one doctor is available per shift.

CSPMM effectively managed to achieve the following, see [Figure 10].

- First, at least 1 doctor was on duty on every shift; hence the doctor demand for every shift was met by the developed system.
- Second, a streamlined and flexible work environment has been created by effectively scheduling doctors and ensuring that the created schedule is fair.
- Third, doctor preferences were taken into consideration as system constraints. Taking doctor preferences into account will help improve their satisfaction and help improve their morale.
- Last, overtime was reduced by effectively scheduling the available resources.

7.3 Recommendations and Future Research

The following further recommendations should be considered to improve the CSPMM. First, creating an optimisation protocol, e.g. simulated annealing to minimise the search area and help to find the best solution out of all the possible solutions that can be generated under the same system specifications (user inputs). The protocol can also be used to eliminate cycling. Second, adding more constraints to the programme and make the system flexible so that the constraints are user-defined, e.g. the number of off days can be defined to the user as being 2 instead of 1 off day in a weekly period. Third, using filtering algorithms to eliminate infeasible solutions and only find the best solution amongst the feasible solutions. Last, conducting a cost-benefit analysis

in order to determine the cost savings of automating the scheduling process. In general, it is important to find how multiple techniques or approaches can be integrated to create schedules as reported by [40, 41] that multiple pertinent approaches can result in accruing breakthrough benefits.

7.4 Limitations of the Study

First, CSPMM assumed that doctors have the same efficiency despite that in practice, doctors have different capabilities [9]. Second, the developed CSPMM did not account for doctors with varying levels of experience. All doctors were assumed to have the same level of expertise. Third, a uniform case was considered, i.e. weekdays (Monday to Friday) were scheduled the same as the weekends (Saturday and Sunday) regarding scheduling rules such as the number and length of shifts and the minimum number of doctors per each shift. Fourth, all staff members (doctors) were treated as permanent staff, i.e. there were no part-time doctors. Fifth, fewer doctors (i.e. 20) were used for testing.

References

[1] Taifa, I. W., & Desai, D. A. (2017). User requirements customization and attractive quality creation for design improvement attributes. *International journal for quality research*, *11*(1).

[2] Taifa, I. W. (2016). *Integration of quality function deployment (QFD) and ergonomics principles in product design improvement. Case study: student desk at engineering college* (Master`s dissertation, Ahmedabad: Gujarat Technological University).

[3] Puente, J., Gómez, A., Fernández, I., & Priore, P. (2009). Medical doctor rostering problem in a hospital emergency department by means of genetic algorithms. *Computers & industrial engineering*, *56*(4), 1232-1242.

[4] Cheng, B. M. W., Lee, J. H. M., & Wu, J. C. K. (1997). A nurse rostering system using constraint programming and redundant modeling. *IEEE Transactions on information technology in biomedicine*, *1*(1), 44-54.

[5] Trilling, L., Guinet, A., & Magny, D. Le. (2007). Nurse scheduling using integer linear programming and constraint programming. *12th IFAC symposium on information control problems in manufacturing INCOM 2006* (pp. 651-656). Saint-Etienne, France: Elsevier.

[6] Banet, D. (2010). *Heuristic scheduling for clinical physicians* (Master thesis,University of Louisville). Retrieved from https://ir.library.louisville.edu/etd/66/

[7] Leksakul, K., & Phetsawat, S. (2011). Nurse scheduling using genetic algorithm. *Mathematical problems in engineering*. http://dx.doi.org/10.1155/2014/246543

[8] Chen, J. G., & Yeung, T. W. (1993). Hybrid expert-system approach to nurse scheduling. *Computers in nursing*, *11*(4), 183-190.

[9] Kuo, Y. H. (2014). Integrating simulation with simulated annealing for scheduling physicians in an understaffed emergency department. *HKIE transactions*, *21*(4), 253-261.

[10] De Villiers, M. R., & De Villiers, P. J. T. (2004). Doctors' views of working conditions in rural hospitals in the Western Cape. *South African family practice*, *46*(3), 21-26.

[11] Leung, J. Y. (Ed.). (2004). *Handbook of scheduling: algorithms, models, and performance analysis*. CRC Press.

[12] Carter, M. W., & Lapierre, S. D. (2001). Scheduling emergency room physicians. *Health care management science*, *4*(4), 347-360.

[13] Beaulieu, H., Ferland, J. A., Gendron, B., & Michelon, P. (2000). A mathematical programming approach for scheduling physicians in the emergency room. *Health care management science*, *3*(3), 193-200.
[14] Wang, T., Meskens, N., & Duvivier, D. (2015). Scheduling operating theatres: Mixed integer programming vs. constraint programming. *European journal of operational research*, *247*(2), 401-413.
[15] Marriott, K., Stuckey, P. J., & Stuckey, P. J. (1998). *Programming with constraints: an introduction*. MIT press.
[16] Rousseau, L. M., Pesant, G., & Gendreau, M. (2000). A hybrid algorithm to solve a physician rostering problem. *Second workshop on integration of AI and OR techniques in constraint programming for combinatorial optimization problems*. Paderborn, Germany.
[17] Cangini, G. (2000). *A constraint programming local search algorithm for physician scheduling*. Université de Montréal, Centre de recherche sur les transports.
[18] Trilling, G. (1998). *Automatic scheduling of doctors on call for the Côte-des-Neiges hospital in Montreal*. University of Montreal, Center for Transport Research.
[19] Bourdais, S., Galinier, P., & Pesant, G. (2003, September). HIBISCUS: A constraint programming application to staff scheduling in health care. *International conference on principles and practice of constraint programming* (pp. 153-167). Springer, Berlin, Heidelberg.
[20] Gendreau, M. (2003). An introduction to tabu search. *Handbook of metaheuristics* (pp. 37-54). Springer, Boston, MA.
[21] Labbé, S. (1998). *Automated scheduling for doctors in emergency rooms* (Master thesis, University of Montreal). Retrieved from http://biblos.hec.ca/biblio/memoires/m1998no56.pdf
[22] Cantera, I. E. B. (2001). *The preparation of working hours of emergency physicians resolved using taboo research* (University of Montreal). Retrieved from https://www.collectionscanada.gc.ca/obj/s4/f2/dsk3/ftp04/MQ60888.pdf
[23] Lo, C. C., & Lin, T. H. (2011, July). A particle swarm optimization approach for physician scheduling in a hospital emergency department. *Seventh international conference on natural computation (ICNC)* (pp. 1929-1933). IEEE.
[24] Eberhart, R. C., & Shi, Y. (1998, March). Comparison between genetic algorithms and particle swarm optimization. *International conference on evolutionary programming* (pp. 611-616). Springer, Berlin, Heidelberg.
[25] Trilling, L., Guinet, A., & Le Magny, D. (2006). Nurse scheduling using integer linear programming and constraint programming. *IFAC proceedings volumes*, *39*(3), 671-676.
[26] Acme Express Inc. (n.d.). *Who's on call?* Retrieved Jul 15, 2013, from https://docsscheduler.net/workspace.aspx?
[27] ByteBloc. (n.d). *Emergency providers scheduling system*. Retrieved Jul 15, 2013, from https://www.bytebloc.com/
[28] Tangier. (n.d.). T*he # 1 emergency medicine scheduling software*. Retrieved Jul 15, 2013, from http://www.tangiersoftware.com/about
[29] Cerner. (n.d). *Clairvia physician scheduler*. Retrieved Jul 15, 2013, from https://klasresearch.com/review/cerner-clairvia-physician-scheduler/1742
[30] Gendreau, M., Ferland, J., Gendron, B., Hail, N., Jaumard, B., Lapierre, S., ... & Soriano, P. (2006, August). Physician scheduling in emergency rooms. *International conference on the practice and theory of automated timetabling* (pp. 53-66). Springer, Berlin, Heidelberg.
[31] Taifa, I. W., & Desai, D. A. (2015). Quality function deployment integration with kano model for ergonomic product improvement (classroom furniture)-a review. *Journal of multidisciplinary engineering science and technology (JMEST)*, *2*(9), 2484-2491.
[32] Taifa, I. W., Desai, D. A., & Bulsara, N. M. (2018). The development of an ergonomically designed product through an integrated product team approach. *International journal of occupational safety and ergonomics*, (just-accepted), 1-32.
[33] Knauth, P. (1993). The design of shift systems. *Ergonomics*, *36*(1-3), 15-28.

[34] Knauth, P. (1996). Designing better shift systems. *Applied ergonomics*, *27*(1), 39-44.
[35] The South African Department of Labour. (n.d). *Basic guides for basic conditions of employment*. Retrieved Aug 1, 2013, from http://www.labour.gov.za/legislation/acts/basic-conditions-of-employment/basic-conditions- of-employment-act-and-amendments
[36] HPCSA. (n.d). *Basic conditions of employment act*. Retrieved May 12, 2013, from http://www.hpcsa.co.za/downloads/psychology/industrial_psychology/act_basic_conditions_ of_employment_2007.pdf
[37] Munassar, N. M. A., & Govardhan, A. (2010). A comparison between five models of software engineering. *International journal of computer science issues (IJCSI)*, *7*(5), 94.
[38] Sheu, P. C. Y. (n.d). *Software lifecycle models*. Retrieved Dec 6, 2018, from http://www.nada.kth.se/~karlm/prutt05/lectures/prutt05_lec6.pdf
[39] National Instruments Corporation. (n.d). *Lifecycle models*. Retrieved from http://zone.ni.com
[40] Taifa, I. W., & Desai, D. A. (2015). A review and gap analysis on integration of quality function deployment and ergonomics principles for product improvement (classroom furniture). *Industrial Engineering journal*, *VIII*, *12*, 16-25.
[41] Taifa, I. W., & Desai, D. A. (2016). Student-defined quality by kano model: a case study of engineering students in India. *International journal for quality research*, *10*(3).