Nurse scheduling problem

**Project idea**

The nurse scheduling problem (NSP) is the operations research problem of finding an optimal way to assign nurses to shifts, typically with a set of hard constraints which all the valid solutions must follow, and a set of soft constraints which define the relative quality of valid solutions.

Solutions to the nurse scheduling problem can be applied to constrained scheduling problems in other fields. The nurse scheduling problem involves the assignment of shifts and holidays to nurses. Each nurse has their own wishes and restrictions, as does the hospital. The problem is described as finding a schedule that both respects the constraints of the nurses and fulfills the objectives of the hospital. Conventionally, a nurse can work

3 shifts because nursing is shift work:

• day shift • night shift • late night shift

In this problem we must search for a solution satisfying as many wishes as possible while not compromising the needs of the hospital.

There are two types of constraints:

1. Hard constraints: if this constraint fails then the entire schedule is invalid.

2. Soft constraints: it is desirable that these constraints are met but not meeting them does not make the schedule invalid.

Some examples of constraints are:

• A nurse does not work the day shift, night shift and late-night shift on the same day (for obversions).

• A nurse may go on a holiday and will not work shifts during this time.

• A nurse does not do a late-night shift followed by a day shift the next day.

The Nurse scheduling problem (NSP) represents a difficult class of multi-objective optimization problems consisting of several interfering objectives between the hospitals and individual nurses. Several constraint-based optimization techniques have been proposed to solve automated nursing scheduling problems in an acceptable computation time but most of these techniques are characterized by premature convergences which inhibit optimal global solution.

The NSP is a staff scheduling problem that intends to assign a set of nurses to work shifts to maximize hospital benefit by considering a set of hard and soft constraints like allotment of duty hours, hospital regulations, and so forth. This nurse scheduling is a delicate task of finding combinatorial solutions by satisfying multiple constraints. The Nurse Scheduling Problem (NSP) is a combination of optimization problem and important management functions performed by nurses who directly affected the hospital services and the patient care. Staff scheduling is the process of constructing work timetables encoding for staff in order to satisfy the demand for services.

**Main functionalities**

**1. Calculate the minimum and maximum numbers of shifts assigned to a given nurse in each week.**

**2. calc the hours worked per week (shift =8).**

**3. days worked consecutively in month.**

**4. days off consecutively in month.**

**Similar applications in the market**

1.Connecteam (Desktop application/Mobile application)

2.10to8 (Desktop application)

3.Deputy (Mobile application)

4.Nurse grid (Mobile application)

5.eSchedule (Web application)

**Academic publications relevant to the idea**

Genetic Algorithm for a Nurse Scheduling Problem

Computers & Operations Research, 31(5), pp 761-778, 2004.

Uwe Aickelin

School of Computer Science University of Nottingham NG8 1BB UK uxa@cs.nott.ac.uk

Kathryn A. Dowsland

Gower Optimal Algorithms Ltd.

5, Whitestone Lane

Newton Swansea

SA3 4UH

UK

Correspondence to U Aickelin

Abstract

This paper describes a Genetic Algorithms approach to a manpower-scheduling problem arising at a major UK hospital. Although Genetic Algorithms have been successfully used for similar problems in the past, they always had to overcome the limitations of the classical Genetic Algorithms paradigm in handling the conflict between objectives and constraints. The approach taken here is to use an indirect coding based on permutations of the nurses, and a heuristic decoder that builds schedules from these permutations. Computational experiments based on 52 weeks of live data are used to evaluate three different decoders with varying levels of intelligence, and four well-known crossover operators. Results are further enhanced by introducing a hybrid crossover operator and by making use of simple bounds to reduce the size of the solution space. The results reveal that the proposed algorithm is able to find high quality solutions and is both faster and more flexible than a recently published Tabu Search approach.

Keywords: Genetic Algorithms, Heuristics, Manpower Scheduling.

Genetic Algorithm for a Nurse Scheduling Problem

1 The Nurse Scheduling Problem

In recent years, Genetic Algorithms (GAs) have become increasingly popular for solving complex optimisation problems such as those found in the areas of scheduling or timetabling. Unfortunately, there is no pre-defined way of including constraints into GAs. This is probably one of their biggest drawbacks, as it does not make them readily amenable to most real world optimisation problems. Some methods for dealing with constraints do exist, notably penalty and repair functions. However, as noted by Michalewicz [1], their application and success is problem specific. Here we use an alternative approach: the genetic operators act on an unconstrained solution space whose elements are converted into solutions by a schedule builder that acts as the decoder for the GA. Our work differs from previous studies using a similar strategy in two ways. First, constraints are used actively to guide the heuristic reducing the reliance on penalty functions. However, the nature of our problem means that it is not possible for the schedule-builder to guarantee feasible solutions and much of our investigation involves finding the best way of balancing the conflicting issues of feasibility and solution cost. Second, we investigate the effectiveness of various different permutation crossover operators for implementations of this type, and consequently introduce a hybrid crossover that combines the desirable features of the best two.

The work described in this paper has two objectives. The first is to develop a fast, flexible solution approach to a nurse rostering problem and second, to add to the body of knowledge on solving constrained problems using GAs. Although there are many published algorithms for nurse scheduling, some characteristics of our particular problem, as outlined in the next section, make these approaches unsuitable for us. An extensive summary of these methods can be found in Hung [2], Sitompul and Randhawa [3] and Bradley and Martin [4]. Examples of approaches based on GAs include Easton and Mansour [5] and Tanomaru [6]. Those who have tackled the problem considered here have met with varying degrees of success. For example, Fuller [7] uses an integer programming approach. She showed that in order to solve the problem reliably within a reasonable amount of computational time it is necessary to use a variety of features that are only available in specialist IP packages. Without these features, around one third of instances required several hours of run-time to find good feasible solutions. Due to the high costs of such software, this is not a practical solution for many hospitals. Dowsland [8] solves the problem using a Tabu Search approach. Her results are impressive in solving instances arising at a particular hospital, but are less good on data with different characteristics.

The approach adopted here was motivated by previous research into a GA solution to the problem. Aickelin and Dowsland [9] show that a straightforward GA implementation is not able to tackle the problem. The addition of problem specific information, in the form of co-evolving sub-populations and a modified fitness function coupled with an intelligent hill-climber, yields reasonable solutions in a very short time. However, solution quality is not consistent and the nature of the modifications means that the approach may not be robust to small changes in problem specification. Thus, there is still room for improvement.

Here we use a different strategy for a GA approach, in which the individuals in the population do not represent direct encodings of solutions. Instead, solutions are obtained via separate decoder heuristics that build solutions from permutations of the list of available nurses using the constraints as guides. These permutations are manipulated by the GA. The advantage of this strategy is that the GA can remain canonical, i.e. it solves an unconstrained problem and does not require a hill-climber or other problem-specific knowledge. Note that penalty functions might still be required if the decoder fails to find a feasible solution. This also makes the algorithm more flexible than the integrated approaches outlined above, as the GA component can remain unchanged if the problem changes, e.g. if there are new constraints or objectives.

The idea of an indirect GA is not new, and similar approaches have been used for other problems, for example by Davis [10] and Palmer and Kershenbaum [11]. However, there is one major difference between our work and theirs: The nurse scheduling problem is not a simple permutation problem like job shop scheduling but has additional constraints and hence our decoders cannot guarantee to always produce feasible solutions. Thus, there is still some work to be done by a penalty function. Our computational experiments are designed to examine this aspect of the implementation, as well as investigating the influence of the choice of heuristic decoder and the crossover operator on solution speed and quality. The result is an algorithm that is still simple but robust enough to cope with this infeasibility. This is achieved by striking a balance between the stochastic elements of the GA and the deterministic nature of the proposed decoder. Our key tools for this are the setting of parameter values, deciding on genetic strategies and finding a suitable decoder function.

2 The Problem

The rostering problem tackled in this paper can be described as follows. The task is to create weekly schedules for wards of up to 30 nurses by assigning one of a number of possible shift patterns to each nurse. These schedules have to satisfy working contracts and meet the demand for a given number of nurses of different grades on each shift, while being seen to be fair by the staff concerned. The latter objective is achieved by meeting as many of the nurses’ requests as possible and considering historical information to ensure that unsatisfied requests and unpopular shifts are evenly distributed. The problem is complicated by the fact that higher qualified nurses can substitute less qualified nurses but not vice versa. Thus scheduling the different grades independently is not possible. Furthermore, the problem has a special day-night structure as most of the nurses are contracted to work either days or nights in one week but not both. However due to working contracts, the number of days worked is not usually the same as the number of nights. Therefore, it becomes important to schedule the ‘right’ nurses onto days and nights respectively. It is the latter two characteristics that make this problem challenging for any local search algorithm as finding and maintaining feasible solutions is extremely difficult. Furthermore, due to this special structure previous nurse scheduling algorithms suggested in the literature cannot be used. Further details of the nurse-scheduling problem can be found in Aickelin and Dowsland [9].

As described in Dowsland and Thompson [12] the problem can be decomposed into three independent stages. The first stage ensures that there are enough nurses to provide adequate cover. The second stage assigns the nurses to the correct number of day or night shifts. A final phase allocates those working on particular day to the early or late shift on that day. Phases 1 and 3 are easily solved using classical optimisation models. Thus, this paper deals with the highly constrained second step.

The numbers of days or nights to be worked by each nurse defines the set of feasible weekly work patterns for that nurse. These will be referred to as shift patterns or shift pattern vectors in the following. For each nurse i and each shift pattern j all the information concerning the desirability of the pattern for this nurse is captured in a single numeric preference cost pij. This was done in close consultation with the hospital and is a weighted sum of the following factors: Basic shift-pattern cost, general day / night preferences, specific requests, continuity problems, number of successive working day, rotating nights / weekends and other working history information.

Patterns that violate mandatory contractual requirements are marked as infeasible for a particular nurse and week. Again, further details can be found in Dowsland and Thompson [12]

The problem can be formulated as an integer linear program as follows.

Indices:

i = 1...n nurse index. j = 1...m shift pattern index. k = 1...14 day and night index (1...7 are days and 8...14 are nights). s = 1...p grade index.

Decision variables:

1 nurse i works shift pattern j

xij =

0 else

Parameters:

n = Number of nurses.

m = Number of shift patterns.

p = Number of grades.

1 shift pattern j covers day / night k a jk =

0 else

1 nurse is i of grade s or higher qis =

0 else

pij = Preference cost of nurse i working shift pattern j.

Ni = Working shifts per week of nurse i if night shifts are worked.

Di = Working shifts per week of nurse i if day shifts are worked.

Bi = Working shifts per week of nurse i if both day and night shifts are worked.

Rks = Demand of nurses with grade s on day / night k.

F(i) = Set of feasible shift patterns for nurse i, where F(i) is defined as

|  |  |  |
| --- | --- | --- |
|  7  ∑a jk =Di   k=1  or  ∑14 jk =Ni  F(i) = a  k=8  or    ∑14 a jk =Bi     k=1 |   ∀j∈day shifts         ∀j∈nightshifts        ∀j∈combined shifts  | ∀i |

Target function: Minimise total preference cost of all nurses

n m

∑∑pij xij → min!

i=1 j∈F( )i

Subject to:

1. Every nurse works exactly one feasible shift pattern:

∑xij = 1 ∀i (1)

j∈F(i)

1. The demand for nurses is fulfilled for every grade on every day and night:

n

∑∑qisa jk xij ≥ Rks ∀k,s (2)

j∈F(i) i=1

Constraint set (1) ensures that every nurse works exactly one shift pattern from his/her feasible set, and constraint set (2) ensures that the demand for nurses is covered for every grade on every day and night. Note that the definition of qis is such that higher graded nurses can substitute those at lower grades if necessary. Typical problem dimensions are 30 nurses of three grades and 411 shift patterns. Thus, the IP formulation has about 12000 binary variables and 100 constraints. Although this is only a moderately sized problem, Fuller [7] shows that some problem instances remain unsolved after hours of computation time on a Pentium II PC (equivalent to the hospital’s hardware) using professional software.

3 Genetic Algorithms

GAs are generally attributed to Holland [13] and his students in the 1970s, although evolutionary computation dates back further (refer to Fogel [14] for an extensive review of early approaches). GAs are stochastic metaheuristics that mimic some features of natural evolution. Canonical GAs were not intended for function optimisation, as discussed by De Jong [15]. However, slightly modified versions proved very successful. For an introduction to GAs for function optimisation, see Deb [16]. Many examples of successful implementations can be found in Bäck [17], Chaiyaratana and Zalzala [18] and others.

In a nutshell, GAs mimic the evolutionary process and the idea of the survival of the fittest. Starting with a population of randomly created solutions, better ones are more likely to be chosen for recombination into new solutions, i.e. the fitter a solution, the more likely it is to pass on its information to future generations of solutions. In addition to recombining solutions, new solutions may be formed through mutating or randomly changing old solutions. Some of the best solutions of each generation are kept whilst the others are replaced by the newly formed solutions. The process is repeated until stopping criteria are met.

However, constrained optimisation with GAs remains difficult. The root of the problem is that simply following the building block hypothesis, i.e. combining good building blocks or partial solutions to form good full solutions, is no longer enough, as this does not check for constraint consistency. To solve this dilemma, many ideas have been proposed of which the major ones (penalty and repair functions) will be briefly outlined in the following. A good overview of these and most other techniques can be found in Michalewicz [1].

Penalty functions try to avoid infeasible solutions by steering the search away from them, whilst repair functions try to ‘fix’ such solutions so that they become feasible. Unfortunately, penalising infeasible solutions is no guarantee that the GA will succeed in finding good feasible solutions, as a highly fit but slightly infeasible solution might dominate the search. If the solution space is dominated by infeasible solutions then finding good feasible solutions with penalty functions alone is unlikely to be successful. Using more complex penalty functions, such as dynamic or adaptive schemes can help, but traditionally most such approaches provide a means of comparing infeasible and feasible solutions during the selection stages rather than dispensing with the problem of satisfying constraints. More promising seem approaches such as used by Beasley and Chu [19] that bias the way the solution space is sampled towards feasible areas. We will use such a scheme in our decoder.

Repairing infeasible solutions also has its drawbacks. Firstly, it is often as difficult to repair an invalid solution as it is to find a good feasible solution. Secondly, repeated repair might lead to a build up of poor material within the population, as there is not enough incentive for development. Finally, repair routines are typically time consuming and it is arguable that this time is better spent on a more direct search of the solution space. Furthermore, the implementation of both penalty and repair functions is highly problem specific and successful applications cannot usually be transferred to other problems. In our case, it is not possible to find a simple, fast repair mechanism that will convert an arbitrary infeasible solution into a feasible one.

The approach presented here is the combination of an indirect GA with a separate heuristic decoder function. The GA tries to find the best possible ordering of the nurses, which is then fed into a greedy decoder that builds the actual solution. One way of looking at this decoder is as an extended fitness function calculation, i.e. the decoder determines the fitness of a solution once it has built a schedule from the permutation of nurses. One advantage of this approach is that all problem specific information is contained within the decoder, whilst the GA can be kept canonical. The only difference from a standard GA is the need for permutation-based crossover and mutation operators as explained for instance in Goldberg [20]. This should allow for easy adaptation to different problems. Furthermore, we will show that such a decoder allows us to use constraints more actively in addition to a background penalty function. These issues will be investigated.

There are similar approaches, reported in the literature, which use decoders that always assemble feasible solutions. This is possible due to the characteristics of the problems studied. Examples are Davis [10] and Fang et al. [21] for job shop scheduling and Podgorelec and Kokol [22] and Corne and Odgen [23] for timetabling problems. In essence, there the decoder is only a schedule builder, whereas in our case it is also a constraint handler that biases the search towards the feasible region. As there is no obvious way of constructing a decoder that only assembles feasible solutions, this research extends previous studies using indirect decoders by allowing infeasible solutions in the solution space. This raises the question as to how the balance between quality and feasibility should be handled in the decoder. We investigate this issue and show that a combination of a stochastic GA and a deterministic and balanced decoder successfully solves the nurse scheduling problem by finding the optimal solution in more than half of all problem instances tackled and being on average within 2% of optimality.

It is also worth noting that our approach does not follow all of the rules suggested by Palmer and Kershenbaum [11] for decoders. Their rules are:

1. For each solution in the original space, there is a solution in the encoded space.
2. Each encoded solution corresponds to one feasible solution in the original space.
3. All solutions in the original space should be represented by the same number of encoded solutions.
4. The transformation between solutions should be computationally fast.
5. Small changes in the encoded solution should result in small changes in the solution itself.

The idea here is that the new solution space should not introduce bias by excluding some solutions, or by representing some by more points than others. In our case rules 1) and 3) are violated, but we argue that such violation is a desirable feature of an indirect representation. The use of a greedy decoder will mean that some solutions are not represented at all. However, these will tend to be lower quality solutions. Similarly some solutions will be represented by more than one permutation, but as these are likely to be feasible or of low cost, the net result is a biased search spending more time in high quality areas of the solution space. However, as our results using different decoders will illustrate, the decoder needs to be designed carefully to ensure that the balance between feasibility and solution quality introduces a useful bias into the solution space, and does not eliminate high quality solutions within, or even close to, the feasibility boundary.

4 An Indirect Approach and the Three Decoders

This section starts with a description of an indirect GA approach whose main feature is a heuristic decoder that transforms the genotype of a string into its phenotype. After discussing the choice of encoding, three different decoders are presented and compared.

The first decision, when using a decoder based GA, has to be what the genotype of individuals should represent.

Here the encodings are required to be of an indirect type, such that they represent an unconstrained problem and the decoder can build a schedule from it. Essentially, this leaves two possibilities in our case: The encoding can be either a permutation of the nurses to be scheduled or a permutation of the shifts to be covered. The former leads to strings of length n for n nurses (max 30) and the decoder would have to assign a shift pattern to each nurse. The latter gives strings of length equal to the number of grades times number of shifts, i.e. 3x14 = 42. In this case, the decoder would assign suitably qualified nurses to shifts. However, this approach leads to difficulties as the nurse preference cost pij is given for full shift patterns, not for single shifts. Thus, we decided to use a permutation of the nurses. This has the further advantage that the multiple-choice constraint set (1) of the integer program formulation is implicitly fulfilled. Hence, this type of encoding requires a less sophisticated decoder.

Having decided to encode the genotypes as permutations of the nurses, a decoder that builds a schedule from this list has to be constructed. This ‘schedule builder’ needs to take into account those shifts that are still uncovered. Additional points to consider are the grades of nurses required, the types and qualifications of the nurses left to be scheduled and the preference cost pij of a nurse working a particular shift pattern. Thus, a good schedule builder would construct feasible or near-feasible schedules, where most nurses work their preferred shift patterns. In the following, we present three decoders for this task, each with a different balance between feasibility and solution quality.

The first decoder is designed to consider only the feasibility of the schedule. It schedules one nurse at a time in such a way as to cover those days and nights with the highest number of uncovered shifts. The second is biased towards solution quality, but includes some aspects of feasibility by computing an overall score for each feasible pattern for the nurse currently being scheduled. The third is a more balanced decoder and combines elements of the first two. These are described in detail below.

The first decoder, referred to as ‘Cover’ decoder in the future, constructs solutions as follows. A nurse works k shifts per week. Usually these are either all day or all night shifts (standard type). In some special cases, they are a fixed mixture of day and night shifts (special type). The first step in the ‘Cover’ decoder is to find the shift with the biggest lack of cover. This will decide whether the nurse will be scheduled on days or nights if the nurse is of the standard type. If there is a tie between, the nurse’s general day / night preference will decide. We then proceed to find the k day or k night shifts with the highest undercover. If the nurse is of the special type, we directly find the k compatible shifts with the highest undercover, taking into account the number of day and night shifts worked by this particular nurse. The nurse is then scheduled to work the shift pattern that covers these k days or nights.

In order to ensure that high-grade nurses are not ‘wasted’ covering unnecessarily for nurses of lower grades, for nurses of grade s, only the shifts requiring grade s nurses are counted as long as there is a single uncovered shift for this grade. If all these are covered shifts of the next lower grade are considered and once these are filled those of the next lower grade. If there is more than one day or night with the same number of uncovered shifts, then the first one is chosen. For this purpose, the days are searched in Sunday to Saturday order. Due to the nature of this approach, nurses’ requests (preference costs pij) cannot be taken into account by the decoder. However, they will influence decisions indirectly via the fitness function (see below for details), which decides the rank of an individual and the subsequent rank-based parent selection. Hence, the ‘Cover’ decoder can be summarised as:

1. Determine type of nurse.
2. Find shifts with corresponding largest amount of undercover.
3. Assign nurse to shift pattern that covers them.

The second decoder, called the ‘Contribution’ decoder, is designed to take account of the nurses’ preferences. It therefore works with shift patterns rather than individual shifts. It also takes into account some of the covering constraints in that it gives preference to patterns that cover shifts that have not yet been allocated sufficient nurses to meet their total requirements. This is achieved by going through the entire set of feasible shift patterns for a nurse and assigning each one a score. The one with the highest (i.e. best) score is chosen. If there is more than one shift pattern with the best score, the first such shift pattern is chosen.

The score of a shift pattern is calculated as the weighted sum of the nurse’s pij value for that particular shift pattern and its contribution to the cover of all three grades. The latter is measured as a weighted sum of grade one, two and three uncovered shifts that would be covered if the nurse worked this shift pattern, i.e. the reduction in shortfall. Obviously, nurses can only contribute to uncovered demand of their own grade or below.

More precisely and using the same notation as before, the score sij of shift pattern j for nurse i is calculated with the following parameters:

dks = 1 if there are still nurses needed on day k of grade s otherwise dks = 0. ajk = 1 if shift pattern j covers day k otherwise ajk = 0. ws is the weight of covering an uncovered shift of grade s. wp is the weight of the nurse’s pij value for the shift pattern.

Finally, (100 - pij) must be used in the score, as higher pij values are worse and the maximum for pij is 100. Note that (-wppij) could also have been used, but would have led to some scores being negative. Thus, the scores are calculated as follows:

3  14 

sij = wp(100− pij )+∑wsqis∑a jkdks 

s=1  k=1 

The ‘Contribution’ decoder can be summarised as follows.

1. Cycle through all shift patterns of a nurse.
2. Assign each one a score based on covering uncovered shifts and preference cost.
3. Choose the shift pattern with the highest score.

The third ‘Combined’ decoder combines the bias towards feasibility of the ‘Cover’ decoder with features of the ‘Contribution’ decoder. It also calculates a score sij for each shift pattern and assigns the shift pattern with the highest score to the nurse, breaking ties by choosing the first such shift pattern. However, in contrast to the ‘Contribution’ decoder, a shift pattern scores proportionally more points for covering a day or night that has a higher number of uncovered shifts. Hence, dks is no longer binary but equal to the number of uncovered shifts of grade s on day k. Otherwise using the same notation as before, the score sij for nurse i and shift pattern j is calculated as before:

3  14 

sij = wp(100− pij )+∑wsqis∑a jkdks 

s=1  k=1 

Thus, the ‘Combined’ decoder can be summarised as follows.

1. Cycle through all shift patterns of a nurse.
2. Assign each one a score proportional to its contribution to uncovered shifts and preference cost.
3. Choose the shift pattern with the highest score.

Finally for all decoders, the fitness of completed solutions has to be calculated. Unfortunately, feasibility cannot be guaranteed, as otherwise an unlimited supply of nurses, respectively overtime, would be necessary. This is a problem-specific issue and cannot be changed. Therefore, we still need a penalty function approach. Since the chosen encoding automatically satisfies constraint set (1) of the integer programming formulation, we can use the following formula, where wdemand is the penalty weight, to calculate the fitness of solutions. Note that the penalty is proportional to the number of uncovered shifts.

∑∑n m pij xij +wdemand ∑∑14 p maxRks −∑∑n m qisa jk xij ; 0 → min!

i= =1 j 1 k= =1 1s  i= =1 j 1 

5 Experiments

This section describes the computational experiments used to test the ideas outlined in the previous section. For all experiments, 52 real data sets as given to us by the hospital are available. Each data set consists of one week’s requirements for all shifts and grade combinations and a list of nurses available together with their preference costs pij and qualifications. The data was collected from three wards over a period of several months and covers a range of scheduling situations, e.g. some data instances have very few feasible solutions whilst others have multiple optima. Unless otherwise stated, to obtain statistically sound results all experiments were conducted as twenty runs over all 52 data sets. All experiments were started with the same set of random seeds, i.e. with the same initial populations. The platform for experiments was a Pentium II PC, which is similar to the hospital’s equipment.

To make future reference easier, the following definitions apply for the measures used in the remainder of this paper: ‘Feasibility’ refers to the mean probability of finding a feasible solution averaged over all data instances and runs. ‘Cost’ refers to the sum of the best feasible solutions (or censored costs as described in the next paragraph) for all data sets averaged over all data sets. Thus, cost measures the unmet requests of the nurses, i.e. the lower the cost the better the performance of the algorithm.

Should the GA fail to find a feasible solution for a particular data set over all twenty runs, a censored cost of 100 is assigned instead. The value of 100 was used as this is more than ten times the average optimal solution value and significantly larger than the optimal solution to any of the 52 data instances. As a result of experiments described in Aickelin and Dowsland [9], the GA parameters and strategies in Table 1 were used for the experiments described in this paper. As can be seen from Table 1, the GA used is a standard generational implementation with fitness-rank-based roulette wheel selection and 10% elitism.

The purpose of the experiments described in this section was threefold. Firstly, to investigate the relationship between the heuristic decoder and the way in which it balances the dual objectives of feasibility and solution cost. Secondly, to compare the results produced by a number of different well-known permutation crossover operators; and thirdly to identify the most promising combination for further enhancement and possibly to identify some ways of achieving this.

The first objective was achieved by running the three decoders on their own and comparing the results. Once these results had been obtained, the ‘Contribution’ decoder was selected for further experimentation to investigate the sensitivity of the results to the way in which ties are broken. The results of the initial comparison are shown in the first three sections of Table 2. These results for the decoders without the GA are shown under the ‘100 random generations’ label and are for 10,000 random nurse permutations, equivalent to approximately 100 generations with a GA or the length of a typical GA run. The results show that the decoders alone are not capable of solving the problem with on average less than 5% of runs finding a feasible solution.

It is interesting to note that on its own the ‘Combined’ decoder is superior to the other two and that between the remaining two the ‘Contribution’ decoder is the better choice for feasibility, but only marginally better in terms of overall cost than the ‘Cover’ decoder. When combined with the four variants of the GA all show a marked improvement in terms of both feasibility and cost. However, only the ‘Combined’ decoder has improved to an acceptable level, producing a large proportion of high quality feasible solutions. The ‘Cover’ decoder produces a reasonable proportion of feasible solutions, but they tend to be of relatively high cost. Although the GA obviously provides some pressure for cost improvement via the fitness function this is not sufficient to overcome the lack of influence of the pij values in this decoder. The results of the ‘Contribution’ decoder are worse than those found by the ‘Cover’ decoder which is surprising as the decoder was designed to be more powerful. In this case, the GA does not seem to be able to guide the search towards permutations yielding feasible solutions. In view of its unexpectedly poor performance, this decoder was the subject of further experiments, aimed at ‘tweaking’ the heuristic in order to improve performance.

The most obvious modification is to change the decoder weights. It could be that the arbitrary choice (set intuitively to w1:w2:w3:wP = 4:2:1:1) conflicts with the GA. A set of experiments to examine this issue (not displayed) showed that the results were poor for any weight setting, although the ratio 8:2:1:1 performed slightly better than the others and was adopted in the following experiments.

A second possible modification is the way in which ties in the scores are broken. Shift patterns are searched in a given order with the first one encountered used in case of ties. In the above experiments, the patterns are ordered using the natural, or lexicographic, ordering, in which patterns involving the earlier days of the week appear first (referred to as ‘Lexico’). This means that at the start of the schedule building when all shifts are equally uncovered there will be a tendency to fill the beginning of the week first. This is likely to have the effect of removing many feasible solutions from the solution space.

In view of this, four further orderings were investigated each designed to give a different balance between a fully randomised selection and bias guided by our knowledge of the problem. The first (‘Rand Order’) randomly shuffles the day and night shifts for each nurse separately, and then uses that order throughout the run. The second (‘Biased’) also uses a random ordering but starts with the day shifts with a 75% probability, reflecting the fact that the ratio between the covering requirements for days and nights is approximately 3:1. The third and fourth orderings order the patterns in increasing pij order (using the lexicographic order to break ties). The third order (‘Rand Cost’) starts from a random point in this order, treating the ordering as a circular list, while the fourth (‘Cheapest’) starts searching at the start of this ordered list, i.e. with the cheapest pattern.

The results of experiments with these orderings are presented in graphical form in Figure 1. The random search orders achieve much better results than the two deterministic ones. Of the three orderings with random starting points, the ‘Biased’ is slightly better than the other two. Full results for this search order are displayed in Table 2 under the ‘Contribution Biased’ heading. The results have improved significantly, to the point where they are not far behind those of the ‘Cover’ decoder - but overall they are still disappointing, especially in view of the fact that the cost element pij is included in this decoder.

One possible explanation could be that although both the ‘Contribution’ and ‘Combined’ decoders produce few feasible solutions on their own, the ‘Combined’ decoder accounts for both the number and magnitude of the under-covered shifts and hence the infeasible solutions tend to be closer to the boundary of the feasible region.

This enables the genetic operators to produce more individuals that score well for both feasibility and cost. Overall, the results show that the choice of decoder is an important factor in the quality of solutions produced. Moreover, while it is not important that the decoder does well on an arbitrary set of permutations it is important that it should give rise to a solution space containing sufficient low cost solutions within or close to the feasibility boundary.

Table 2 also allows a comparison of four commonly used permutation crossover operators in conjunction with the different decoders. Ordered by the quality of results over all decoders, starting with the best, they are PMX (Goldberg and Lingle [24]), uniform order based crossover (Syswerda [25]), C1 crossover (Reeves [26]) and order-based crossover (Davis [27]). The rationale behind these experiments was that each type of crossover has different characteristics in terms of information passed to children regarding the order and position of genes. The percentage of genes which keep the same absolute position from one parent to a child are on average 33% for order based, 50% for C1 and uniform order based and 66% for PMX. Additionally for the order based, C1 and uniform order based crossovers the remaining genes keep the absolute order of the other parent. Hence, the results suggest that the higher the percentage of genes staying in their absolute position the better. Moreover, in the case of two operators with the same percentage, i.e. uniform order based crossover and C1 crossover, the more disruptive and hence flexible uniform crossover performs better. These results are not surprising, in that intuitively the good permutations will be those that schedule ‘difficult’ nurses first, leaving the easy ones until later. However, they do suggest that a crossover that is similar to uniform, i.e. not processing large blocks but keeping a higher proportion of genes in their absolute positions may result in further improvements.

6 Extensions

In the previous section, the ‘Combined’ decoder with PMX crossover was identified as the most promising variant for our nurse-scheduling problem. In this section we consider further modifications / enhancements leading to our final implementation. These are:

* Using the ‘Biased’ in place of the ‘Lexico’ ordering for the ‘Combined’ decoder.
* Fine-tuning the penalty weight used in the fitness function.
* Introducing a new crossover operator.
* Using a simple bound to improve the heuristic in the latter stages of the search.

The more detailed measure of undercover used in the score for the ‘Combined’ decoder means that the undesirable bias resulting from breaking ties with the ‘Lexico’ ordering will have less influence. Nevertheless, there will still be a tendency to allocate personnel to the beginning of the week. Therefore, the randomised orderings may give some improvement. A limited amount of experimentation was carried out to investigate this and indicated that the ‘Biased’ ordering is able to produce slightly better results. Therefore, this was adopted.

However, Even after this improvement, not all runs resulted in feasible solutions. Therefore, fine-tuning the weight of the preference cost wp, was considered worthwhile. Figure 2 shows the outcome of a series of experiments for different values of wp indicated by the x-axis labels. As the graph shows, the results are sensitive to this parameter as it decides the balance between cost and feasibility. The behaviour for variations of wp is as would be expected. If it is set too low, then solutions are very likely to be feasible but are of high cost. If wp is set too high, solution quality rapidly drops due to the bias towards cheap but infeasible solutions. A value of wp = 0.5 gives the best results, sacrificing only a small amount of feasibility for a good improvement in cost.

However, the cost of solutions still leaves some room for improvement.

In an attempt to further improve results, a new type of crossover was introduced. This new crossover operator is inspired by the results found previously and combines the higher number of genes left in their absolute positions (like PMX) and re-ordering of blocks of information (like uniform permutation crossover). This new operator, called parameterised uniform order crossover or PUX for short, works in a similar way to uniform order based crossover as described in Syswerda [25]. However, when creating the binary template, the probability for a one will be equal to a parameter p, similar to standard parameterised uniform crossover [14]. For instance, if p = 0.66, then there will be a 66% chance for a one and hence a 66% chance that a gene will be in the same absolute position as in one of the parents.

Thus, PUX with p = 0.66 has an equal probability of keeping a gene in the same absolute position as PMX. However, PUX has an advantage. Whilst with PMX the remaining 33% of genes were positioned without any reference to the parents, PUX retains the absolute order of these as found in the second parent. In line with other uniform crossover operators, PUX is disruptive in a sense that it does not transmit large chunks of the parents. In order to find the best value for p, experiments were carried out shown in Table 2 under the ‘Combined PUX’ section. The value after each PUX label indicates the percentage used for p. (Note that p = 50% corresponds to the original uniform order based crossover). Particularly for the cost of solutions, results are further improved with an appropriate choice for p. For instance, for p = 0.66 feasibility is as high as for PMX, but solution cost is significantly lower. This suggests that our hypotheses about the necessary qualities of a successful crossover operator for our problem were right.

Another interesting observation is that the higher the value of p the lower the feasibility of solutions. This indicates that a more disruptive crossover, i.e. p = 0.5, is more flexible and has the power to create feasible solutions when the other operators fail. On the other hand, solution cost is best for a medium setting of p, i.e. p = 0.66 or p = 0.8. This shows that for best results a careful balance has to be struck between flexibility and a parameter setting, which allows larger chunks to be passed on more frequently. In our case a setting of p = 0.66 is ideal.

The final enhancement of the indirect GA is based on the intelligent use of bounds. Herbert [28] concludes that results using a GA and a decoder can be improved with a new type of crossover operator. Herbert considers maximisation problems and uses a permutation based encoding with C1 crossover, which makes intelligent use of upper bounds, obtained from the fitness of partial strings. He argues that once a lower bound for the solution has been found only crossover points within that part of the string with an upper bound on fitness greater than the lower bound should be used. This is because the ‘mistakes’ in a particular solution must have happened before such a boundary point.

However, due to the difficulty of defining good upper bounds on partial fitness values of our strings, it is unclear how such a sophisticated operator would work in this case. Therefore, we propose a ‘simple bound’ based on a similar idea. When building a schedule from the genotypes it is easy to calculate the sum of the pij costs so far. Once this sum exceeds a bound, set equal to the best feasible solution found so far, the schedule has ‘gone wrong’. We could now employ backtracking to try to correct this. However, to guarantee that we improve the solution with the actual permutation of nurses at hand, a sophisticated algorithm of exponential time complexity would be necessary. This is outside the scope of this piece of research, but might be an idea for future work.

Instead, we propose a simpler approach. Once a feasible solution of cost C\* has been found, we know that in the optimal solution no nurse i can work a shift pattern j with pij > C\*. We will use this as a rule when assigning shift patterns. In the early stages of optimisation this simple bound is of little use, as C\* >> pij. However, towards the end of the search when good feasible solutions have been found, the simple bound should prevent wasting time on dead-end solutions by making sure shift patterns with pij ≤ C\* are assigned.

Table 2 shows that the use of the simple bound slightly improved the cost of solutions whilst leaving the feasibility unchanged. This can be attributed to those runs where good solutions were found which were then further improved by forcing nurses onto even cheaper shift patterns. An additional benefit of using the simple bound was that average solution time was accelerated from around 15 seconds per single run to less than 10 seconds.

7 Summary of Results

A comparison of final results is shown in Figure 3, with details for the best algorithm in Figure 4. The simple

‘Cover’ decoder and ‘Contribution’ decoder produced mediocre results. Once both decoders were ‘Combined’ (label ‘Combo’), the results rivalled those found with the most sophisticated direct GA including all enhancements [9] (label ‘Direct’). However, once the PUX operator and the simple bounds were employed as well (label ‘PUX’), the results were better than for any direct approach and within 2% of optimality.

A look at Figure 4 shows that 51 out of 52 data sets are solved to or near to optimality and a feasible solution is found for the remaining data set. The bars above the y-axis represent solution quality, with the black bars showing the number of optimal solutions, and the total bar height showing the number of solutions within three units of the optimal value. The value of three was chosen as it corresponds to the penalty cost of violating the least important level of requests in the original formulation. Thus, solutions this close to the optimum would certainly be acceptable to the hospital. The bars below the axis represent the number of times out of 20 that the run terminated without finding a single feasible solution. Hence the less the area below the axis and the more above, the better the performance.

Once the experiments had identified the best set of decoders and parameters, the final indirect GA was tested on a series of modifications to the problem specifications supplied by the hospital and on additional randomly generated data sets. In all cases, the modifications were easily added and the algorithm performed well. Moreover, results found for these new data sets were better than those for Dowsland’s Tabu Search [8]. Thus, we feel our aim of creating a robust evolutionary algorithm for this problem has been achieved.

8 Conclusions

This paper presents an alternative algorithm for solving a nurse-scheduling problem in the form of a GA coupled with a decoding routine. In comparison to the previous ‘direct’ GA approach described in Aickelin and Dowsland [9] this has two advantages: Firstly, the GA solves an unconstrained problem leaving the constraint handling to the decoder that uses them to directly bias the search rather than in penalty functions alone. Secondly, all problem specific knowledge is held in the decoder routine, thus the algorithm can be quickly adapted to changes in problem specification. The overall results are better than those found by previous evolutionary approaches with a more flexible implementation than Tabu Search.

The nature of this type of decoder means that some of the conditions for an ideal transformation between the solution space and search space, as suggested by Palmer and Kershenbaum [11] cannot be satisfied. A comparison of three decoders showed that only when the decision making process struck the right balance between feasibility and quality good overall solutions were obtained. For problems that are more complex, one can imagine that setting these weights by hand will no longer be satisfactory. Consequently, some of our current research looks into possibilities of the algorithm finding its own weights intelligently with early results reported in Aickelin [29].

The role of the crossover operator was also shown significant. Again, this appears to be a question of balance – in this case between disrupting long sub-strings and inheriting absolute positions from the parents. The PUX operator was introduced to facilitate such a balance and the results of experiments with this operator suggested that more disruptive operators helped feasibility, but that too much disruption affected solution quality.

The quality of the final results, together with the fact that the indirect GA approach proved to be more flexible and robust than Tabu Search, makes the indirect GA a good choice to solve the nurse-scheduling problem. Its central idea of changing the problem into a permutation problem and then building solutions with a separate decoder can be applied to all constrained scheduling problems. Thus given this success, experiments with similar approaches on other difficult problems, in scheduling and other application areas is an interesting area for further research.

Bibliography

1. Michalewicz Z. A Survey of Constraint Handling Techniques in Evolutionary Computation Methods.

Proceedings of the 4th Annual Conference on Evolutionary Programming 1995: 135-155.

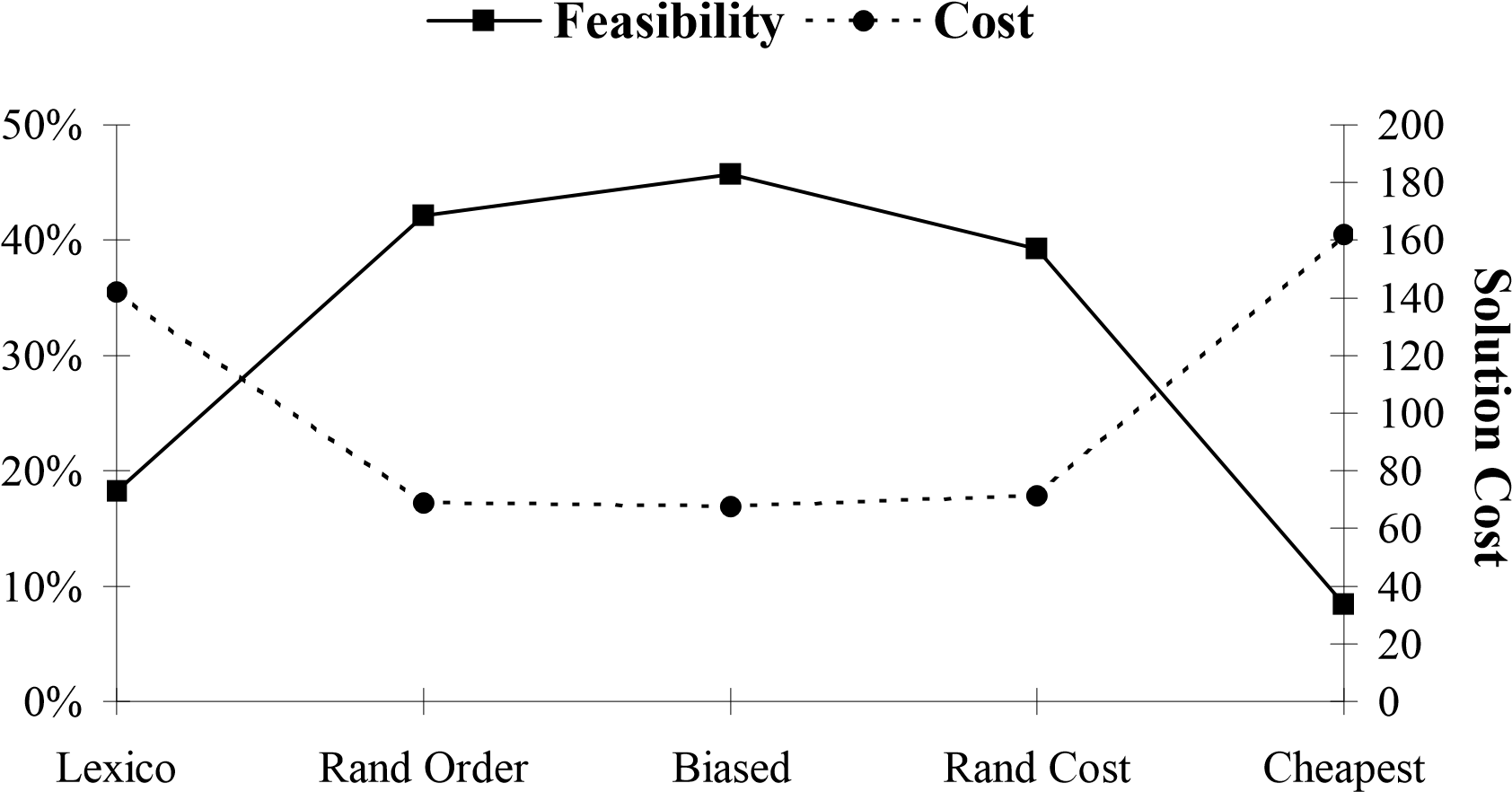
1. Hung R. Hospital Nurse Scheduling. Journal of Nursing Administration 1995; 1:21-23.
2. Sitompul D and Randhawa S. Nurse Scheduling Models: A State-of-the-Art Review. Journal of the Society of Health Systems 1990; 2:62-72.
3. Bradley D and Martin J. Continuous Personnel Scheduling Algorithms: A Literature Review. Journal of the Society for Health Systems 1990; 2:8-23.
4. Easton F and Mansour N. A Distributed Genetic Algorithm for Employee Staffing and Scheduling Problems. In Forrest S. Editor. Proceedings of the Fifth International Reference on Genetic Algorithms, San Mateo: Morgan Kaufmann Publishers, 1993:360-367.
5. Tanomaru J. Staff Scheduling by a Genetic Algorithm with Heuristic Operators. Proceedings of the IEEE Reference on Evolutionary Computation 1995; 456-461.
6. Fuller E. Tackling Scheduling Problems Using Integer Programming. Master Thesis, University of Wales Swansea, United Kingdom, 1998.
7. Dowsland KA. Nurse Scheduling with Tabu Search and Strategic Oscillation. European Journal of Operational Research 1998; 106:393-407.
8. Aickelin U and Dowsland KA. Exploiting problem structure in a Genetic Algorithms approach to a nurse rostering problem. Journal of Scheduling, Vol. 31, pp 139-153, 2000.
9. Davis L. Adapting Operator Probabilities in Genetic Algorithms. In Schaffer J. editor. Proceedings of the Third International Reference on Genetic Algorithms and their Applications, San Mateo: Morgan Kaufmann Publishers, 1989:61-67.
10. Palmer C and Kershenbaum A. Representing Trees in Genetic Algorithms. Proceedings of the First IEEE International Reference on Evolutionary Computation 1994:379-384.
11. Dowsland KA and Thompson JM. Nurse Scheduling with Knapsacks, Networks and Tabu Search. Journal of the Operational Research Society 2000; 825-833.
12. Holland J. Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press, 1976. [14] Fogel D. Evolutionary Computation: The Fossil Record. IEEE Press, 1998.
13. De Jong K. Genetic Algorithms are NOT Function Optimisers. In Whitley D. Editor. Foundations of Genetic Algorithms 2. San Mateo: Morgan Kaufmann Publishers, 1993:5-17.
14. Deb K. Genetic Algorithms for Function Optimisation. Genetic Algorithms and Soft Computing 1996:431.
15. Bäck T. Applications of Evolutionary Algorithms. 5th Edition, Dortmund, Germany, 1993.
16. Chaiyaratana N, Zalzala A. Recent Developments in Evolutionary and Genetic Algorithms: Theory and Applications. In Fleming P, Zalzala S. editors. Genetic Algorithms in Engineering Systems 2: Innovations and Applications. IEEE Proceedings, Letchworth: Omega Print & Design, 1997:270-277.
17. Beasley J and Chu P. A Genetic Algorithm for the Set Covering Problem, European Journal of Operational Research 94, 392-404, 1996.
18. Goldberg D. Genetic Algorithms in Search, Optimization and Machine Learning, Reading: AddisonWesley, 1989.
19. Fang H-L, Ross P and Corne D. A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling and Open Shop Scheduling Problems. In Forrest S. editor. Proceedings of the Fifth International Reference on Genetic Algorithms, San Mateo: Morgan Kaufmann Publishers, 1993: 375382.
20. Podgorelec V and Kokol P. Genetic Algorithm Based System for Patient Scheduling in Highly Constrained Situations. Journal of Medical Systems 1997; 21:417-427.
21. Corne D and Odgen J. Evolutionary Optimisation of Methodist Preaching Timetables. In Burke E, Carter M. editors. Practise and Theory of Automated Timetabling II, Heidelberg: Springer Verlag, 1998:143155.
22. Goldberg D and Lingle R. Alleles, Loci, and the Travelling Salesman Problem. In Grefenstette J. editor. Proceedings of the First International Reference on Genetic Algorithms and their Applications, Hillsdale New Jersey: Lawrence Erlbaum Associates Publishers, 1985:154-159.
23. Syswerda G. Schedule Optimisation Using Genetic Algorithms. In Davis L. editor. Handbook of Genetic Algorithms, London: International Thomson Computer Press, 1996:335-349.
24. Reeves C. Hybrid Genetic Algorithms for Bin-Packing and Related Problems. Annals of OR 1996; 63:371-396.
25. Davis L. Applying Adaptive Algorithms to Epistatic Domains. Proceedings of the ninth International Joint Reference on Artificial Intelligence 1985; 1:162-164.
26. Herbert E. Genetic Algorithms for Bin Packing Problems. PhD Dissertation, University of Wales Swansea, United Kingdom, 1998.
27. Aickelin U. An Indirect Genetic Algorithm for Set Covering Problems, Journal of the Operational Research Society, Special Issue on Local Search, in print, 2002.

|  |  |
| --- | --- |
| Parameter / Strategy | Setting |
| Population Size | 100 |
| Population Type | Generational |
| Initialisation | Random |
| Selection | Proportional to Fitness-Rank |
| Crossover | Two parents, Order-Based |
| Bit Swap Mutation Probability | 1.5% |
| Replacement Strategy | Keep 10% Best |
| Stopping Criteria | No improvement for 30 generations |
| Penalty Weight | 20 |

Table1: Parameters and strategies used for the indirect GA and nurse scheduling following [9].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Decoder | Type of indirect GA | Time [s] | Feasibility | Cost |
| Cover | 100 random generations | 29.2 | 2.7% | 191.3 |
| C1 Crossover | 19.7 | 42.1% | 68.7 |
| Order Crossover | 17.4 | 39.2% | 69.7 |
| Uni Crossover | 17.4 | 45.4% | 63.3 |
| PMX Crossover | 18.5 | 46.4% | 61.7 |
| Contribution | 100 random generations | 27.3 | 3.2% | 190.2 |
| C1 Crossover | 18.9 | 13.9% | 154.2 |
| Order Crossover | 17.2 | 12.7% | 155.9 |
| Uni Crossover | 16.5 | 14.9% | 151.5 |
| PMX Crossover | 17.1 | 16.6% | 144.9 |
| Combined | 100 random generations | 25.2 | 5.2% | 171.0 |
| C1 Crossover | 14.0 | 92.3% | 15.0 |
| Order Crossover | 14.4 | 88.4% | 19.0 |
| Uni Crossover | 15.0 | 97.4% | 14.1 |
| PMX Crossover | 15.3 | 96.6% | 14.3 |
| Contribution  Biased | 100 random generations | 27.1 | 4.2% | 142.1 |
| C1 Crossover | 19.9 | 39.2% | 71.2 |
| Order Crossover | 19.4 | 37.3% | 74.5 |
| Uni Crossover | 19.7 | 42.1% | 68.7 |
| PMX Crossover | 18.9 | 45.6% | 67.5 |
| Combined  Biased PUX | PUX 50% | 15.0 | 97.4% | 14.1 |
| PUX 66% | 14.6 | 96.6% | 10.0 |
| PUX 80% | 13.9 | 96.0% | 10.2 |
| PUX 90% | 13.8 | 95.4% | 14.3 |
| PUX 66% and simple bound | 9.3 | 96.6% | 9.4 |
| Summary | Indirect GA | 9.3 | 96.6% | 9.4 |
| Direct GA [9] | 14.9 | 90.9% | 10.8 |
| TABU [8] | 30 | 100% | 8.8 |
| IP [7] | >6000 | 100% | 8.7 |

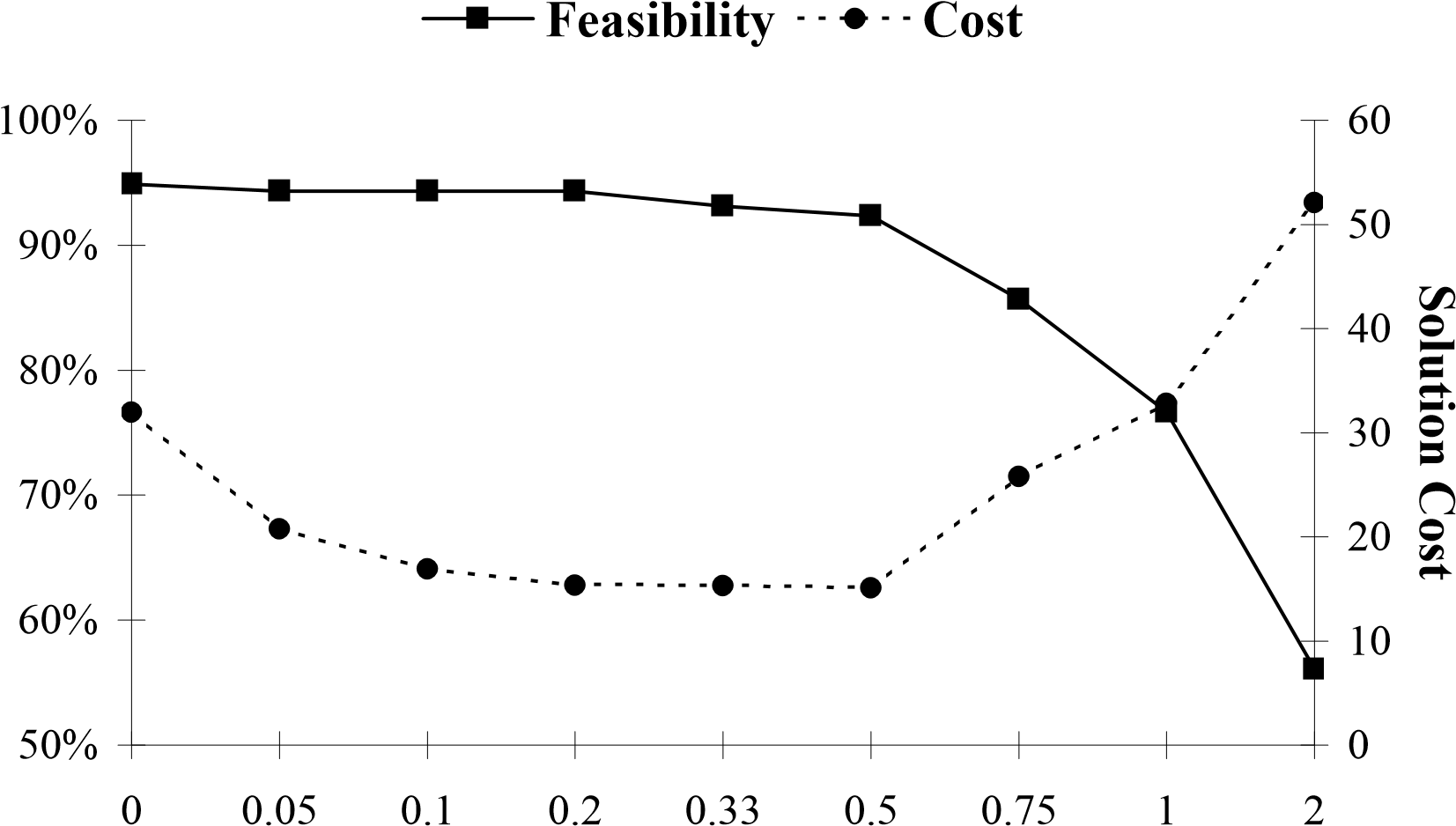
Table 2: Comparison of Results. Lower cost, respectively higher feasibility is better.



Feasible Solutions

# Variations in Shift Pattern Ordering

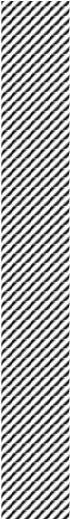
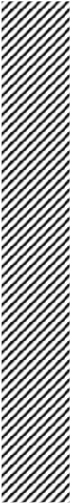
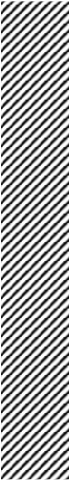
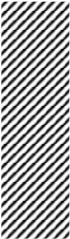
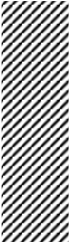
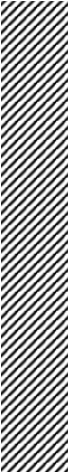
Figure 1: Effect of different search orders for the ‘Contribution’ decoder. Lower cost, respectively higher feasibility is better.



Feasible Solutions

# Preference Weight

Figure 2: Variations of preference weight for the ‘Combined’ decoder and its effect on solution quality. Lower cost, respectively higher feasibility is better. Note that the x-axis is not to scale.



0

20

40

60

80

100

Feasibility [%] /

Solution Cost



Feasibility



Cost

Direct Cover Contrib Combo PUX Tabu

Type of Optimisation Algorithm

Figure 3: Comparison of results of various heuristics approaches to the nurse-scheduling problem. Lower cost, respectively higher feasibility is better.

Genetic Algorithm



No. infeasible No. optimal No. within 3



Figure 4: Detailed results for all 52 data sets using the enhanced indirect GA. The bars show the number of infeasible, good feasible and optimal solutions out of 20.

Hindawi Publishing Corporation



Mathematical Problems in Engineering Volume 2014, Article ID 246543, 16 pages http://dx.doi.org/10.1155/2014/246543

*Research Article*

**Nurse Scheduling Using Genetic Algorithm**

# Komgrit Leksakul1,2 and Sukrit Phetsawat2 *1*

*Excellence Center in Logistics and Supply Chain Management, Chiang Mai University, Chiang Mai 50200, Thailand 2*

*Department of Industrial Engineering, Faculty of Engineering, Chiang Mai University, Chiang Mai 50200, Thailand*

Correspondence should be addressed to Komgrit Leksakul; komgrit@eng.cmu.ac.th

Received 20 May 2014; Revised 20 October 2014; Accepted 23 October 2014; Published 27 November 2014

Academic Editor: Purushothaman Damodaran

Copyright © 2014 K. Leksakul and S. Phetsawat. This is an open access article distributed under the Creative Commons Attribution License,whichpermitsunrestricteduse,distribution,andreproductioninanymedium,providedtheoriginalworkisproperlycited.

This study applied engineering techniques to develop a nurse scheduling model that, while maintaining the highest level of service, simultaneously minimized hospital-staffing costs and equitably distributed overtime pay. In the mathematical model, the objective function was the sum of the overtime payment to all nurses and the standard deviation of the total overtime payment that each nurse received. Input data distributions were analyzed in order to formulate a simulation model to determine the optimal demand for nurses that met the hospital’s service standards. To obtain the optimal nurse schedule with the number of nurses acquired from the simulation model, we proposed a genetic algorithm (GA) with two-point crossover and random mutation. After running the algorithm, we compared the expenses and number of nurses between the existing and our proposed nurse schedules. For January 2013, the nurse schedule obtained by GA could save 12% in staffing expenses per month and 13% in number of nurses when compare with the existing schedule, while more equitably distributing overtime pay between all nurses.

# Introduction

In order to succeed in organization management, one of the important factors that should be taken into consideration is human resource management within the organization for maximum efficiency at all times. This will enable the organization to always drive the mission to the target successfully with the highest efficiency and effectiveness. For personnel management within hospitals, scheduling the work of nurses is one factor that is important and difficult to manage for maximum efficiency due to the uncertainty of the number of patients each day, which causes difficulty in managing nursing staff to adequately and appropriately provide services to patients. If there are too many nurses, the hospital will likely unnecessarily waste the budget. However, if there are too few nurses, the hospital will not have enough nurses for all the patients or each nurse may receive or have to take on excessive workload [1]. Due to the high complication associated with nurse scheduling problems, using people to schedule may cause errors easily, such as the task not being done with high effectiveness and the process taking longer times.

Approaches in the 1970s and 1980s addressed a number of problem formulations and solution techniques. A goal in many studies was to provide support tools to reduce the need for manual construction of nurse scheduling. Some studies [2–6] addressed the problem of determining staff levels and skills based on the numbers of patients and their medical need. Further advances [7–9] were made in applying linear and mixed integer programming and network optimization techniques for developing nurse scheduling. The numbers of researches have included a mix of heuristic and simulation techniques in an attempt to deal with more complex nurse scheduling. As real world problems are immense and deal with many constraints heuristics and recently metaheuristic such as simulated annealing (SA), tabu search (TA), and genetic algorithms (GA) have been developed to generate high quality nurse schedules in an acceptable computation time. Reference [10] proposed TA for constructing nurse scheduling whose objective is to ensure that enough nurses are on duty at all times while taking account of individual preference and requests for days off in a way that is seen to treat all nurses fairly. Recently, GA has been applied to staff scheduling in different application areas such as transportation systems [11], health care systems [12], emergency services, ambulance and fire brigade, call centers, and many other service organizations.

In this paper, we intended to solve nurse scheduling problem by combining mathematical model, computer simulation, and GA. Nonlinear integer model was formulated. The average total overtime payment and fairly payment to all the nurses were considered to minimize which maintains the standard of service level. When mathematical models are applied to solve the problems, the objective function becomes the sum of the overtime payment of all the nurses and the standard deviation of the total overtime payment that each nurse received. This study aimed to create the most appropriate nurse scheduling model by applying the simulation technique to analyze the distribution of the data. Thereafter, a simulation model was created to determine the appropriate demand for nurses with no more than 15% of patients waiting longer than the average service time, as well as those with more than 25% of patients waiting, and, with the application of GA coding in Matlab program, to calculate the best work schedule from the most appropriate demand for nurses acquired from the simulation. Then the expenses and the number of nurses from the new work schedule were compared with the actual expenses of the work schedule and the previous number of nurses.

# Problem Analysis and Formulation

*2.1. Cause and Problem Analysis of Nurse Scheduling.* Currently, one hospital is using manpower planning for nurse scheduling. This method involves workload and productivity efficiency measure per hour of work on average by having 20 nurses in the outpatient department who are rotated for work in the following five departments: (1) department of surgery (SUR), (2) department of internal medicine (MED), (3) department of eye, ear, nose, and throat (EENT), (4) department of pediatrics (PED), and (5) department of obstetrics and gynecology (OBG). There are two types of nurses’ working schedules: 8 hours per day (08.00–16.00 hours) and 12 hours per day (8.00–20.00 hours), with fixed workload each day, and the working time cannot be changed at this stage. This causes problems while allocating the holidays in order to meet the needs of the nurses [13, 14].

Nurse scheduling at this hospital uses only one format which assigns the same number of nurses to take care of each department every week. This format of scheduling is not consistent with the number of patients treated each day and does not bring into consideration factors related to service

levels and queuing systems in nurse scheduling. It is not possible to allocate similar rates of overtime payment for all nurses. The researcher emphasizes the importance of an appropriate and effective nurse scheduling model for ensuring the least cost to the hospital while maintaining the level of service of the hospital and being fair in paying overtime to all the nurses at similar rates by using simulation and optimization method to create the most appropriate nurse scheduling.

*2.2. Mathematical Modeling and Demand Function.* We created the objective and the limitation equations under various conditions based on the information obtained from the hospital by developing an integer programming mathematical model with assignment problem [15, 16]. The preliminary data were set in the following conditions:

{{1, when nurse 𝑖 works on date 𝑗

𝑥𝑖𝑗𝑘𝑙 = {{ at time 𝑘 in department 𝑙 (1)

{0, otherwise,

where 𝑖 = 1, 2, 3, . . . , 20 (nurse), 𝑗 = 1, 2, 3, . . . , 7 (seven working days: Monday, Tuesday, Wednesday, . . ., Sunday), 𝑘 = 1, 2 (two working shifts: regular working hours (08.00– 16.00 hours) and overtime hours (16.00–20.00 hours)), 𝑙 = 1, 2, 3, 4, 5 (department of surgery (SUR), department of internal medicine (MED), department of eye, ear, nose, and throat (EENT), department of pediatrics (PED), and department of obstetrics and gynecology (OBG)), 𝑁 = total number of nurses, SOT = sum of all the overtime payments that nurses receive, SD = standard deviation of the total overtime payment that each nurse receives, AVG = average of total overtime payment that each nurse receives, OTD𝑖𝑗𝑙 = additional overtime payment that nurse 𝑖 receives on the working date 𝑗 in the department 𝑙, OTW𝑖 = overtime payment that nurse 𝑖 receives in a week, OTS𝑖 = total overtime payment that nurse 𝑖 receives, and Demand𝑗𝑘𝑙 = demand for nurses on date 𝑗 at time 𝑘 in the department 𝑙.

The objective equation includes two purposes.

1. The expenses of the hospital are minimal.
2. The standard deviations of the total overtime payments of the nurses are the most similar.

Doing a calculation to find the answers for the problem patterns with multiple objectives would be highly complicated, and it would be difficult to find an answer. To make it easier to calculate, two equations were combined together. However, since both the equations are in different units, to include the two together, it is necessary to change them into the same unit by making these two equations in the form of a percentage of the total. Weighting the importance of the target functions to 𝑤 percent and (1−𝑤) percent, respectively, willmakeupthesumofthehospitalcostpercentageequation, and the standard deviation of the total overtime payment that each nurse receives can be estimated as follows:

SOT SD

Min 𝑍 = 𝑤 × ( ) + (1 − 𝑤) × ( ) , (2) AvgSOT AvgSD

where 𝑤 = weight adjusted, AvgSOT = sum of average total overtime payment that every nurse receives in that particular month, and AvgSD = standard deviation of the total overtime payment that each nurse receives in that particular month. Constraints equation includes the following.

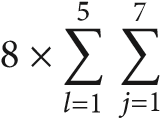
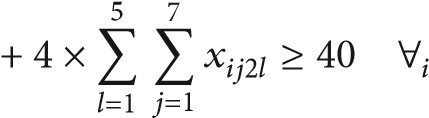
1. The number of nurses must be adequate to meet the needs of the patients in each working period and day:

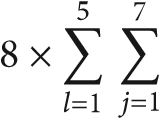
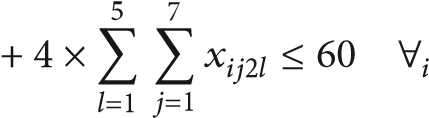
𝑁

∑ 𝑥𝑖𝑗𝑘𝑙 ≥ Demand𝑗𝑘𝑙 ∀𝑗 ∀𝑘 ∀𝑙. (3)

𝑖=1

1. One nurse must work at least 40 hours per week (4) but fewer than 60 hours per week (5):

𝑥𝑖𝑗1𝑙  (4)

𝑥𝑖𝑗1𝑙 . (5)

1. All nurses must be selected to work (6) and work not over six days per week (7):

|  |  |  |
| --- | --- | --- |
| 7 |  |  |
| ∑ 𝑥𝑖𝑗𝑘𝑙 ≥ 1  𝑗=1 | ∀𝑖 ∀𝑘 ∀𝑙 | (6) |

𝑥𝑖𝑗𝑘𝑙 ≤ 6 ∀𝑖 ∀𝑘 ∀𝑙. (7)

1. In a day, one nurse can work only one period in one department:

|  |  |  |
| --- | --- | --- |
| 𝑥𝑖𝑗11 + 𝑥𝑖𝑗12 + 𝑥𝑖𝑗13 + 𝑥𝑖𝑗14 + 𝑥𝑖𝑗15 ≤ 1 | ∀𝑖 ∀𝑗, | (8) |
| 𝑥𝑖𝑗21 + 𝑥𝑖𝑗22 + 𝑥𝑖𝑗23 + 𝑥𝑖𝑗24 + 𝑥𝑖𝑗25 ≤ 1 | ∀𝑖 ∀𝑗. |  |

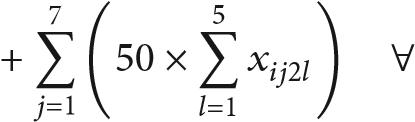
Equations related to expenses are as follows.

When nurses work for more than 40 hours in a week, they will receive an overtime payment of 650 Baht per eight hours or 81.25 Baht per hour, and if within 16.00–20.00 hours, they will receive 50 Baht extra overtime payment per working day:

5 7 5 7

OTS𝑖 = 81.25 × [(8 × ∑ ∑ 𝑥𝑖𝑗1𝑙 + 4 × ∑ ∑ 𝑥𝑖𝑗2𝑙) − 40]

[ 𝑙=1 𝑗=1 𝑙=1 𝑗=1 ]

𝑖,

∑𝑁𝑖=1 OTS𝑖

AVG = ,

𝑁

𝑁

SOT = ∑ OTS𝑖,

𝑖=1

SD = √ ∑𝑁𝑖=1 (OTS𝑖 − AVG)2 .

𝑁

(9)

After that, the least number of nurses (𝑖) is determined in order to be scheduled under existing limitations.

Proposed mathematical model with objective function (2) subject to constraints (3)–(9) was verified by simple problem, as shown in Table 1. With the conventional exact solution approach, the nurse scheduling was constructed for two departments, SUR and MED, and reported by Gantt chart, as shown in Table 2. In Table 2, ∗ and # represent the schedule of SUR and MED, respectively. Lingo software also reported that the mathematical model is nonlinear model with integer variables.

# Building Simulation to Find AppropriateDemand for Nurses

In this case, since a service criterion system has not been in use in the hospital, the demand for nurses as in the present may not be an appropriate demand for finding the least cost work schedule. The researcher examined an appropriate demand for nurses (Demand𝑗𝑘𝑙) by creating a simulation using the Arena program to determine the appropriate demand for nurses in each department and in all the periods of each day from the original hospital data. It was found that, on average, 34.08% or 54.87 patients per week had waited longer than 25% of the total average service time. Therefore, the researcher aimed to bring about an improvement at the initial state itself by creating a simulation under the service criteria that set the limitation as only 15% of the patients on average did not have to wait longer than 25% of the total average service time.

*3.1. Data Collection of Patient Service and Nursing Service Duration.* For the data collection of each department at each period of each day (starting from when the patients register until they wait to see a doctor), the researcher chose to use the data as each day in the month to represent day in that month. Collecting data were done by the quality assurance section in case study hospital; they defined the outpatient based on the combination of appointments and walk-ins. Interarrival time for regular and overtime hour including service time has been collected for two years in winter season which has the largest number of outpatients. The interarrival time for regular and overtime periods were separately collected, due to significant static data. However, no significant difference was noticed for service time. Then, the data were classified by day in a week and analyzed for their statistics distribution for three categories.

*3.2. Data Distribution Analysis.* The data were analyzed by using a suitable dispersion, input analyzer, in the Arena program, and by selecting the distribution with the highest 𝑃

value using the significance level (𝛼) at 0.05 (95% confidence), as shown in Table 3. The 𝑃 value is a key concept in the approach of Ronald Fisher, where he uses it to measure the weight of the data against a specified hypothesis and as a guideline to ignore data that does not reach a specified significance level.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 1: Demand for nurses: verify case study.   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Department | Period |  |  | Hospital demand (person) |  |  | |  |  | Mon. | Tue. | Wed. Thu. Fri. | Sat. | Sun. | | SUR | Regular | 1 | 1 | 2 1 2 | 1 | 2 | |  | Overtime | 1 | 2 | 1 2 1 | 1 | 1 | | MED | Regular | 2 | 1 | 2 1 1 | 1 | 1 | |  | Overtime | 1 | 1 | 1 2 1 | 2 | 2 | |

*3.3. Simulation Modeling.* We created a simulation modeling starting from when the patient registered at the service department based on interarrival time and service time statistics distribution reported in Table 3. Because these statistics distributions consider the arrival of appointment and walk-in patients then this simulation model can take into account the dynamics of walk-in patients. Beginning with statistics distributions patients’ arrival, patients had registered at the service department. According to patient’s symptom, they had moved to required department counter and waited for medical history file. To parallel with file searching and coming, patients had queued and waited for nurse service. Nurses served the fundamental examination such as weight, blood pressure, temperature measure, presymptom examined and finally recorded all measure data in the file after it arrived. Service time distributions recorded and fitted by statistic approach in Table 3 were used as the data set in process box. After that, we saved the percentage information of those patients whose waiting time exceeded the service time limitation, as shown in Figure 1.

Then we adjusted the number of nurses (resource) in the capacity column manually, as shown in Table 4, until the percentage of the patients waiting for more than the limit (average and half-width) in each department was less than 15%, according to what we wanted in every section, as shown in Table 5. After that, we processed 100 replications.

From Table 4, it can be observed that the departments of SUR, MED, and EENT appointed two people, PED appointed three people, and OBG appointed one person to work in their departments.

From Table 5, we can see that all the departments with the percentage of patients waiting for more than the limit (average and half-width) do not exceed 15%. The EENT, MED, OBG, PED, and SUR departments had the percentages of patients waiting for longer than the limit (average plus positive half-width) at 0.7918%, 2.0505%, 5.0424%, 2.7194%, and 5.6773%, respectively. We considered only the positive half-width because we would like to know the maximum percentage of patients waiting.

*3.4. Simulation Model Verification.* In model verification, we compared the results between real situation which occurred in this hospital and the simulation model. Average number of patients receiving service, average number of patients waiting longer than average service time, and average number of patients waiting less than average service time were proposed for preforming simulation model efficient. The 100 times with eight-hour simulation were conducted for all five outpatients departments. The results were reported in Table 6. In simulation row, we reported the average number plus/minus with their corresponding half-width.

Table 6 reported that the OBG got the highest % error in terms of average number of patients receiving service due to the low 𝑃 value reported in Table 3. With this error, however, we can adopt this simulation model generated by the arena program for real case application.

After model verification, manual adjustment of the resource capacity was done and selected under hospital top management service policy. The service level policy was set as no more than 15% of patients waiting longer than the average service time. Then, the appropriate resource (number of nurses) from simulation was shown for each department in Table 7. This number of nurses will be used as the data for nurses demand (Demand𝑗𝑘𝑙) in the mathematical model for obtaining the optimal nurse schedule. Even though, to obtain the optimal nurse schedule with conventional approach always consumes computational runtime and hardware implementation. Genetic algorithm was proposed to solve and obtain an acceptable solution.

# Programming for Genetic AlgorithmNurse Scheduling

We coded a genetic algorithm program for use in the shift scheduling of nurses with the Matlab program by using the following steps.

*4.1. Chromosome Format and Chromosome Encoding.* A chromosome format uses a binary number system, and it is divided into three levels, as shown in Figure 2, where level 1 is a nurse at 1, 2, 3, . . . , 𝑁; level 2 is the working days Monday, Tuesday, Wednesday, . . ., Sunday; level 3 is the number of 8 bit code.

*4.2. Chromosome Encoding and Chromosome Decoding.* To encode the chromosome, we used the binary coded decimal system which is a system that uses the 8 bit binary code instead of the general code we have set up. We can encode and decode the chromosome, as shown in Table 8, where

1. NDA = not working during regular hours,
2. DA1 = working during regular hours at Division 1,
3. DA2 = working during regular hours at Division 2,(iv) DA3 = working during regular hours at Division 3,
4. DA4 = working during regular hours at Division 4,
5. DA5 = working during regular hours at Division 5,
6. NOT = not working overtime,
7. OT1 = working overtime at Division 1,
8. OT2 = working overtime at Division 2,(x) OT3 = working overtime at Division 3,

(xi) OT4 = working overtime at Division 4, (xii) OT5 = working overtime at Division 5.

Table3:Distributionpatientsinterarrivalandservicetime.

DayDepartment

RegularhourOvertimehour

Servicetime

𝑃

valueAvg.time

Interarrivaltime

𝑃

valueInterarrivaltime

𝑃

value

Monday

SUR

−

0.5+

EXPO

(8.52)0.333

−

EXPO(10.7)0.2250.5+WEIB

(7.11,1.85)

0.5+

>

0.7506.84

MED

−

0.5+

(5.39)0.654

EXPO

−

0.5+EXPO(7.68)0.616POIS(4.13)0.0514.13

PED

−

0.5+WEIB(9.51,1.14)0.113

−

0.5+EXPO(12.3)0.0750.5+EXPO(6.5)0.0787.00

EENT

−

0.5+EXPO(9.54)0.4785.5+EXPO(16.5)0.1710.5+EXPO(4.58)0.6155.08

OBG

−

0.5+WEIB(14.6,1.54)0.46511.5+EXPO(22.5)0.228TRIA(2.5,3,19.5)0.2449.10

Tuesday

SUR

−

0.5+

(10.5,1.2)0.334

WEIB

−

0.5+WEIB(9.87,0.925)0.7020.5+18

∗

BETA(0.774,1.25)0.4817.40

MED

−

0.5+WEIB(6.32,1.33)0.510

−

0.5+EXPO(9.13)0.7460.5+LOGN(4.12,5.32)0.1094.43

PED

−

EXPO

(5.6)0.607

0.5+

−

0.5+ERLA(4.98,2)0.5261.5+EXPO(8.7)0.06610.20

EENT

−

0.5+GAMM(6.63,1.11)0.063

−

0.5+EXPO(10.5)0.0570.5+18

∗

BETA(0.409,1.23)0.4274.98

OBG

−

0.5+GAMM(7.59,1.3)0.1451.5+EXPO(30.1)0.125TRIA(2.5,5.96,19.5)0.4219.32

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

Sunday

SUR

−

(9.7,12.6)

LOGN

0.5+

>

0.750

−

0.5+EXPO(14.4)0.0680.5+17

∗

BETA(0.76,1.06)0.5817.37

MED

−

0.5+GAMM(9.47,1.16)0.2831.5+EXPO(22.4)0.105UNIF(0.5,18.5)0.5898.76

PED

−

0.5+WEIB(5.12,0.841)0.197

−

0.5+EXPO(10.3)0.7330.5+EXPO(4.5)0.2105.00

EENT

−

(

LOGN(10.5,16.6)0.349UNIF

0.5+

−

0.5,19.5)0.4380.5+19

∗

BETA(0.898,1.56)0.4897.44

OBG

−

0.5+LOGN(13.1,20.7)0.6730.5+EXPO(17.5)0.097POIS(6.35)0.1266.35

Service Level

SUR

Process SUR

1

Decide

MoreMean 1

LowerMean 1

0

0

True

False

0

0

Dispose 1

IT SUR

8

hr

02 01

Service Level

MED

Process MED

Decide 2

MoreMean 2

LowerMean 2

0

0

True

False

0

0

Dispose 2

IT MED

8

hr

02 01

Service Level

PED

Process PED

Decide 3

MoreMean 3

LowerMean 3

0

0

True

False

0

0

Dispose 3

IT PED

8

hr

02 01

Service Level

EENT

Process EENT

Decide 4

MoreMean 4

LowerMean 4

0

0

True

False

0

0

Dispose 4

IT EENT

8

hr

02 01

Service Level

OBG

Process OBG

Decide 5

MoreMean 5

LowerMean 5

0

0

True

False

0

0

Dispose 5

IT OBG

8

hr

02 01

Figure 1: The simulation model created by the Arena program.

Table 4: The example for adjusting the number of nurses in the Arena program.

|  |  |  |
| --- | --- | --- |
| Name | Type | Capacity |
| 1 Resource SUR | Fixed capacity | **2** |
| 2 Resource MED | Fixed capacity | **2** |
| 3 Resource PED | Fixed capacity | **3** |
| 4 Resource EENT | Fixed capacity | **2** |
| 5 Resource OBG | Fixed capacity | **1** |

Table 5: Average and half-width values based on number of nurses adjusted.

|  |  |  |
| --- | --- | --- |
| Output | Average | Half-width |
| EENT | **0.4418** | **0.35** |
| MED | **1.2505** | **0.80** |
| OBG | **3.4024** | **1.64** |
| PED | **1.6794** | **1.04** |
| SUR | **4.4273** | **1.25** |

*4.3. Creating Initial Population.* Each nurse’s chromosome can be randomly generated for the initial state. Then check whether it is in the scope of primary goal or not, with working hours fewer than 60 hours. If that chromosome is not in the scope of the primary goal, again random search until each nurse’s chromosome completes for the entire length and all the nurse’s chromosomes are generated to reach the number of population specified.

*4.4. Crossover and Mutation.* Crossover is an important operator which combines the good properties of both parents in order to possibly yield new better children chromosomes [17, 18]. As usual, the simple crossover operator (one-point crossover) consists of randomly choosing a crossover point and then recombining the pieces of a pair of chromosomes to form two new chromosomes. The simple crossover is compatible with the random keys encoding, though it generally fails to preserve the permutation when dealing with natural encoding. Hence, for natural encoding, special crossover operators must be used [19]. Roulette wheel selection (RWS), ranking selection (RS), tournament selection (TS), partially matched crossover (PMX), order crossover (OX), and cycle crossover (CX) are evaluated by comparing the performance of GA’s operators on university course timetabling problem [20]. PMX, OX, and CX operators require two crossover points. Given two parent chromosomes, 𝐴 and 𝐵, child chromosome 𝐴󸀠 will inherit form 𝐴 the subsequence between these two points and child chromosome 𝐵󸀠, form 𝐵, the respective subsequence. The elements of 𝐴󸀠 and 𝐵󸀠 outside the two points are copied from the other parent chromosome, while trying to preserve its position under the PMX operator or by trying to preserve its order under the OX operator. A comparative analysis of PMX, CX, and OX crossover operators for solving travelling salesman problem (TSP) was reported in [21]. The experimental results show that the PMX crossover outperforms the CX and OX crossover operator in TSP with 25 numbers of cities. For our problem, simple test was conducted and we found PMX provided a better solution

Day

1

1

2

2

3

3

4

4

N

···

···

Mo

Tu

We Th Fr Sa Su

Number of nurses

Main chromosome

11

1

1

00

0

0

Binary code

Level

1

Level

2

Level

3

N ∗ 7 ∗ 8

Figure 2: The chromosome format.

than one-point crossover; specifically, one-point crossover cannot take into account all constraints (3)–(8).

After applying crossover, the mutation operator acts on the pairs of chromosomes. Although mutation occurs infrequently in nature, it is believed to be an important driving force for evolution. The mutation is adopted to allow for the introduction of new chromosome into the population and is effective to escape from a local optimum. The simple mutation operates by randomly changing its value with a given probability used in our experiments.

*4.5. Evaluation of Fitness Value.* The fitness value can be evaluated by using two criteria, as follows.

Criterion one: a chromosome is a feasible solution.

(i) Fitness value is the objective function that is processed by (2).

Criterion two: a chromosome is a nonfeasible solution.

(i) Fitness value is equal to 1000 units.

*4.6. Chromosome Selection.* We selected the chromosome by using roulette.

*4.7. Elite Preserve Strategy.* The previous chromosomes with high fitness values were replaced by the new chromosomes with the lower fitness values. Then, they were used as the initial population for the next iteration.

*4.8. Termination Criteria.* The condition to stop seeking answers was that when the required number of solutions fully meets the numbers of iteration specified; we stop the examination process immediately.

# Experiment Design to Determine OptimalConfiguration in Genetic Programming

In this section, we introduced design of experiment (DOE) technique for screening the effected parameters and parameters setting guideline for genetic algorithm solution search

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 6: Comparison between real situation and simulation model.   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Department | Calculated result | AVG number of  patients receiving service | AVG number of patients waiting  longer than AVG time | AVG number of  patients waiting less than AVG time | Percentage of patients waiting longer than  AVG time | | | SUR | Real situation  Simulation | 59  61.63 ± 1.63 | 0.2 0.19 ± 0.36 | 58  60.51 ± 1.66 | 0.338%  0.196 ± 0.32% | | |  | % error | 4.27% | 5.26% | 4.33% | — | | | MED | Real situation  Simulation | 96  98.79 ± 1.91 | 0  0 ± 0 | 98  97.94 ± 1.91 | 0% 0% | | |  | % error | 2.90% | 0% | 0.06% | — | | | PED | Real situation  Simulation | 55  57.58 ± 1.48 | 0  0 ± 0 | 53  56.69 ± 1.46 | 0% 0% | | |  | % error | 4.69% | 0% | 6.96% | — | | | EENT | Real situation  Simulation | 53  54.46 ± 1.76 | 18.2 16.73 ± 4.14 | 34  36.64 ± 3.46 | 34.34%  19.99 ± 5.12% | | |  | % error | 2.75% | 8.07% | 7.76% | — |  | | OBG | Real situation  Simulation | 34  38.4 ± 0.85 | 0  0 ± 0 | 35  37.7 ± 0.84 | 0% 0% |  | |  | % error | 12.94% | 0% | 7.71% | — |  | |  |  | Table 7: Demand for nurses obtained from simulation model. | | |  |  | | Department | Period | Hospital demand (person)  Monday Tuesday Wednesday Thursday Friday | | | Saturday | Sunday | | SUR  MED  PED | Regular | 1. 2 2 2 2 3 3 3 3 3 2. 3 3 3 3 | | | 2  2 3 | 2  2 3 | | EENT |  | 2 2 2 2 2 | | | 2 | 2 | | OBG |  | 2 2 2 2 2 | | | 2 | 2 | | SUR  MED  PED | Overtime | 2 2 2 2 2 3 2 2 2 2  2 2 2 2 3 | | | 1  2  2 | 2  2  2 | | EENT |  | 2 2 2 2 2 | | | 2 | 2 | | OBG |  | 1 1 1 2 1 | | | 1 | 1 | |

machine. We conducted the experiment by fixing current number of nurses at 18 with the low level nurses’ demand in each department. All three expecting parameters were set by two levels, maximum and minimum.

*5.1. Determining Factors and Factor Values in Each Level.* Factors can be chosen for the experiment as follows:

1. population number,
2. percent crossover rate,
3. percent mutation rate.

We used the factors in Table 9 to construct the experimental design matrix, full factorial design (2𝑘) (8), and then conducted the experiment by using three center points per block with all the three replicates. We also fixed the number of maximum cycles to 400 cycles. This was because the research revealed that the number of the calculated cycles had high stability. The cycles also took a long time to calculate and, so, may cause inconvenience in their actual use. The number of nurses, 18 in total, was fixed at the beginning for the experimental design, and the results are shown in Table 10 which includes the use of the data for the demand for nurses from the first week of January.

*5.2. Effect and Coefficient Analysis and Model Adequacy Checking.* We used the experimental results to analyze the effects and coefficients by using the Minitab program in order to find the factors affecting the expected responses.

Figure 3 shows that the data were normally and consistently distributed without any tendency and can be used to analyze further results.

Table 11 shows that the correlation of the data is linear because the center pt. is greater than 0.05 and the 𝑃 values of the 𝐴, 𝐶, 𝐴𝐶, 𝐵𝐶, and 𝐴𝐵𝐶 factors are less than 0.05, which is significant for the experiment. It was concluded that the 𝐴

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 8: Example for encoding binary coded decimal.   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Decimal | Binary | Regular hour | Overtime hour | Code | Type | Demand 1 | Demand 2 | Hours | | 0 | 00000000 | NDA | NOT | NDA, NOT | Type 0 | 0 | 0 | 0 | | 1 | 00000001 | DA1 | NOT | DA1, NOT | Type 1 | DA1++ | 0 | 8 | | 2 | 00000010 | DA2 | NOT | DA2, NOT | Type 2 | DA2++ | 0 | 8 | | 3 | 00000011 | DA3 | NOT | DA3, NOT | Type 3 | DA3++ | 0 | 8 | | ... | ... | ... | ... | ... | ... | ... | ... | ... | | 255 | 11111111 | NDA | NOT | NDA, NOT | Type 255 | 0 | 0 | 0 |   Residual plots for obj.    8  6  4  2  0  26  24  22  20  18  16  14  12  10  8  6  4  2  0.010  0.005  0.000  Observation order  Histogram  Versus order  Residual  −0.010  −0.005  0.010  0.005  0.000  Residual  −0.010  −0.005  Frequency  Figure 3: The residual plots for objective function. |

Table 9: Selected experimental factors and factor values in each level.

|  |  |  |  |
| --- | --- | --- | --- |
| Factor |  | Level |  |
|  | Low |  | High |
| (1) Population number (𝐴) | 10 |  | 50 |
| (2) Percent crossover rate (𝐵) | 5 |  | 95 |
| (3) Percent mutation rate (𝐶) | 5 |  | 95 |

and 𝐶 factors were significant at the confidence level of 0.05, as shown in Figure 4.

The 𝑅-Sq. value ranges from 0 to 1 or from 0% to 100%. If the 𝑅-Sq. value is very close to 1, then it indicates that the simulation can explain the different variables more properly and accurately. Table 12 shows that the 𝑅-Sq. value is equal to 99.48%, which is very high. This means that the data can explain dependent variables properly and can analyze further response optimizer functions.

*5.3. Analysis for Finding out Most Appropriate Results by Using Response Optimizer.* The results were analyzed by using a response optimizer of Minitab, as shown in Figures 5 and 6.

From the results obtained, as shown in Figure 5, this study aims to determine the lowest target equation. As a result, we changed “Goal” to “Minimize” since the target equation was the sum of the hospital expense percentage and the standard deviation of the overtime each nurse received which ranged from 0 to 100. As a result, the target was set to 0 and the upper target to 100.

Figure 6 shows that if we want to configure the genetic program to obtain the lowest target equation, it must be set as shown in Table 13. From the prediction equation, the target equation should be equal to 0.4492 (obj. = 0.4492).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 10: Full factorial design (23) experiment with three center points per block type.   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Run | Std. order | Run order | Center pt. Blocks 𝐴 | | 𝐵 | 𝐶 | Obj. | Time (s) | | 1 | 1 | 1 | 1 1 10 | | 5 | 5 | 0.45892 | 9237.521 | | 2 | 2 | 2 | 1 1 50 | | 5 | 5 | 0.50282 | 57621.902 | | 3 | 3 | 3 | 1 1 10 | | 95 | 5 | 0.47113 | 7904.851 | | ... | ... | ... | ... ... ... | | ... | ... | ... | ... | | 25 | 25 | 25 | 0 1 30 | | 50 | 50 | 0.48018 | 31281.694 | |  |  |  | Table 11: Effect and obj. coefficient analysis results. | | |  |  |  | |  |  |  | Factorial fit: obj. versus 𝐴, 𝐵, 𝐶  Estimated effects and coefficients for obj. (coded units) | | |  |  |  | | Term |  | Effect | Coeff. | SE coeff. | |  | 𝑇 | 𝑃 | | Constant |  |  | 0.48216 | 0.001318 | |  | 365.75 | 0.000 | | 𝐴 |  | 0.05200 | 0.02600 | 0.001318 | |  | 19.72 | 0.000 | | 𝐵 |  | 0.00027 | 0.00013 | 0.001318 | |  | 0.10 | 0.921 | | 𝐶 |  | −0.02134 | −0.01067 | 0.001318 | |  | −8.10 | 0.000 | | 𝐴 ∗ 𝐵 |  | −0.00285 | −0.00143 | 0.001318 | |  | −1.08 | 0.293 | | 𝐴 ∗ 𝐶 |  | −0.00850 | −0.00425 | 0.001318 | |  | −3.22 | 0.005 | | 𝐵 ∗ 𝐶 |  | −0.01395 | −0.00697 | 0.001318 | |  | −5.29 | 0.000 | | 𝐴 ∗ 𝐵 ∗ 𝐶 |  | −0.00974 | −0.00487 | 0.001318 | |  | −3.69 | 0.002 | | Ct. Pt. |  |  | −0.00198 | 0.003955 | |  | −0.50 | 0.623 | |

Table 12: 𝑅-square decision-making coefficients.

𝑆 = 2082.09 PRESS = 152191461

𝑅-Sq. = 99.48% 𝑅-Sq. (pred.) = 98.99% 𝑅-Sq. (adj.) = 99.25%

Table 13: Guideline setting for genetic program from response optimizer function.

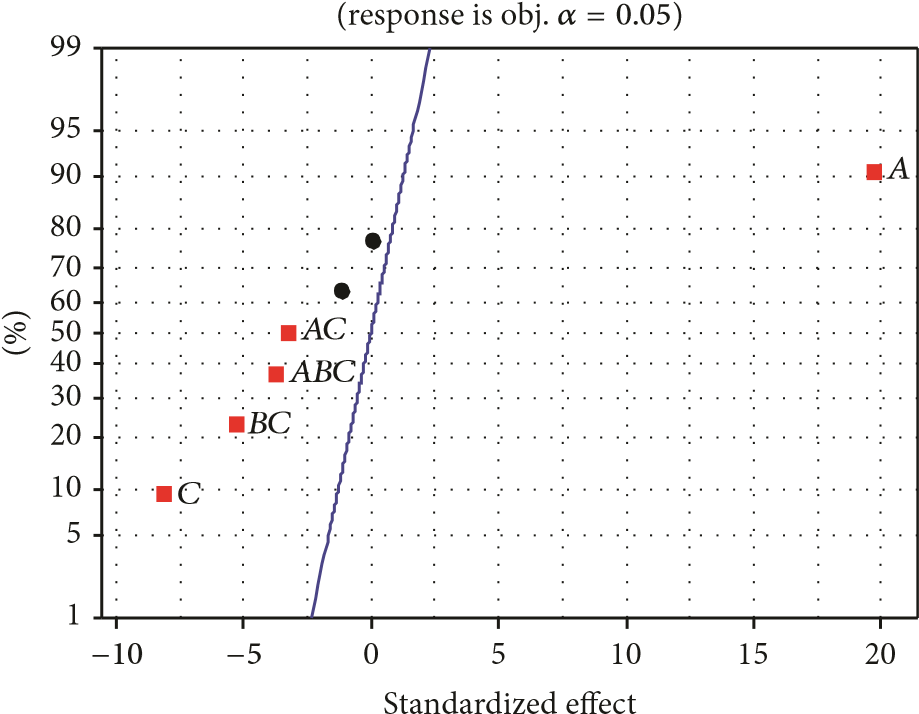
|  |  |
| --- | --- |
| Factor | Level |
| (1) Population number (𝐴) | 10 |
| (2) Percent crossover rate (𝐵) | 95 |
| (3) Percent mutation rate (𝐶) | 95 |

Possible optimal parameters setting may occur at the low level % mutation; we conducted the experiment and reported in Table 14. The reports showed that the objective function is worse when % mutation decreased. The reports also shown that the higher % mutation, the better objective functions.

*5.4. Result Confirmation Experiment.* We experimented to check for accuracy by configuring the genetic program, as shown in Table 12, together with fixing the maximum iterations to the number of nurses at 400 cycles and 18 nurses, respectively, as shown in Table 15.

Table 15 shows that the actual target value (obj.) is actually better than (i.e., less than) 0.63% of the predicted value and canbeusedpracticallyandactuallydecreasedthetargetequation (obj.).

Normal plot of the standardized effects



Effect type Factor Name

Not significant A A

Significant B B C C

Figure 4: The normal plot of the standardized effects (response is obj., alpha = 0.05).

# Change in Number of Nurses (𝑖) in Optimal Scheduling under Constraint Equation

We changed the number of nurses (𝑖) which was an index in the constraint equation in order to find out the least

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 14: Experimental results obtained at low level % mutation.   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Max. iteration | Population | % crossover | % mutation | Obj. | Time (s) | | 1500 | 10 | 95 | 95 | 0.44442 | 8,049 | | 1500 | 10 | 95 | 1 | 0.48580 | 10,161 | | 1500 | 10 | 95 | 0.1 | 0.50286 | 10,523 | | 1500 | 10 | 95 | 0.01 | 0.53612 | 10,117 |   Table 15: Target equation (obj.) predicted by Minitab program and compared with actual target equation (obj.).   |  |  | | --- | --- | | Target equation (obj.) | Comparison of target equation (obj.) percentage | | Predicted value Actual value | Actual value compared with predicted value | | 0.4492 0.4429 | 0.63% | |

Table 16: Calculation results with reduction in number of nurses (𝑖) at 𝑤 = 0.6.

|  |  |  |
| --- | --- | --- |
| Number of nurses (𝑖) | Target equation (obj.) | Calculating time (hour) |
| 21 | 0.5241 | 7.67 |
| 20 | 0.3959 | 6.03 |
| 19 | Infeasible | Infeasible |
| 18 | Infeasible | Infeasible |



Figure 5: The response optimizer configuration of obj. response.

number of nurses (𝑖) who were able to schedule the work under the restrictions by using genetic programming and by gradually reducing the number of nurses (𝑖) down to the final value at which the program could calculate the answer. The calculation was performed at 𝑤 = 0.6 in Table 16.

Table 16 shows that the least number of nurses (𝑖) which could schedule work under the restrictions was 20. Based on 20 nurses, amount of overtime payment and standard deviation of overtime payment were calculated and reported in Table 17. Therefore, number of working hours for each nurse, number of nurses in each department, and nurse’s scheduleplanweregeneratedandreportedasshowninTables 18 and 19.

From Table 17, it can be observed that the target equation (obj.) of the responses obtained was equal to 0.3959. The sum of the overtime payment of all nurses was 16,600 Baht per

Cur.

High

Low

0.99551

Optimal

minimum

Obj.

0.99551

desirability

Composite

5.0

95.0

5.0

95.0

10.0

50.0

50.0

400.0

95.0

10.0

400.0

95.0

y = 0.4492

d = 0.99551

A

C

B

D

D

Figure 6: The results of the data analysis using the response optimizer of the obj. response.

week. The standard deviation of overtime payment that each nurse received was 214.843. For fair comparison, we have to convert the real overtime payment and standard deviation for current 18 nurses to 20 nurses. Then, we can report that the sum of the total overtime payment and the standard deviation of the overtime payment that each nurse received which is obtained from the model were less than those of the old working schedule, at 17,991 Baht per week and 882.369, respectively.

Table 18 shows that all the nurses worked according to the conditions specified; that is, one nurse must work at least 40 hours per week, as in (4), but must work fewer than 60 hours per week, as in (5), all would be selected to work, as in (6), and the duration of work must not exceed six days a week, as in (7).

From Table 19, we can see that the number of nurses working in shifts each day was more than the demand for nurses (Demand𝑗𝑘𝑙), as shown in Table 6, which is according to (3), and the specified service criteria (on average, 15% of the patients waiting for services should not wait longer than the average service time of 25%) because the number of nurses working in shifts was more than the demand.

Tostudytheeffectofweightadjusted(𝑤),morenumerical experiments were conducted by GA coded in Matlab at 400

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 17: Calculating results by 20 nurses (𝑖).  Target equation Calculating time Sum of overtime payment Standard deviation of total overtime  (obj.) (hour) of nurses (Baht) payment that each nurse received   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0.3959 |  | 6.03 16,600 | | | | | 214.843 |  | |  |  | Table 18: Number of working hours calculated from 20 nurses (𝑖). | | | | |  |  | | Number of nurses |  |  |  | Hours/person (hour) | |  |  |  | |  | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday | Total | | 1 | 12 | 12 | 0 | 4 | 0 | 8 | 12 | 48 | | 2 | 4 | 8 | 12 | 4 | 8 | 12 | 0 | 48 | | 3 | 8 | 8 | 8 | 0 | 12 | 12 | 0 | 48 | | 4 | 12 | 0 | 12 | 12 | 12 | 4 | 0 | 52 | | 5 | 8 | 12 | 8 | 4 | 0 | 0 | 12 | 44 | | ... | ... | ... | ... | ... | ... | ... | ... | ... | | 20 | 8 | 12 | 4 | 0 | 12 | 8 | 4 | 48 | |  |  | Table 19: Number of | nurses working | in shifts calculated | by 20 nurses | (𝑖). |  |  | | Department | Period | Supply (person)  Monday Tuesday Wednesday Thursday Friday | | | | | Saturday | Sunday | | SUR  MED  PED | Regular | 1. 2 2 2 2 3 3 3 4 3 2. 3 4 3 3 | | | | | 2 2  4 | 2  2 3 | | EENT |  | 2 3 2 2 2 | | | | | 2 | 2 | | OBG |  | 2 2 2 2 2 | | | | | 2 | 2 | | SUR  MED  PED | Overtime | 2 2 2 2 2 3 2 2 2 2  2 2 2 2 3 | | | | | 1  2 3 | 2  2 3 | | EENT |  | 3 2 2 2 2 | | | | | 2 | 2 | | OBG |  | 2 2 3 2 1 | | | | | 1 | 2 | |

Table 20: Effect of weight adjusted to objective function.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Weight adjusted |  |
|  | 𝑤 = 0.9 | 𝑤 = 0.6 | 𝑤 = 0.5 |
| Objective function | 0.5090 | 0.4753 | 0.4823 |
| Sum of overtime payment | 24375 | 24625 | 26575 |
| Standard deviation | 324.898 | 297.675 | 288.509 |

iterations and reported in Table 20. The report was shown as we expected; more weight adjusted less overtime payment simultaneously with higher standard deviation. It means that if management focuses on reducing the overtime payment, it will lead to unfair payment or bias nurses’ schedule plan.

# Comparison of Advantages andDisadvantages of Adaptive Genetic and Optimization Approach

We processed the data to do a comparison between the results of the adaptive genetic approach and the results obtained by the Lingo program using the same data in order to compare the target equation (obj.) and computation runtime, as shown in Tables 21 and 22.

Table 21 shows the results of the processing experiment after it was conducted ten times. The work schedules using 20 and 24 nurses with the calculating cycle of 400 rounds resulted in the best value for the target equations of 0.4361 and 0.4551, respectively. After ten processing experiments, the work schedules for 20 and 24 nurses with the calculating cycle at 1,500 rounds resulted in the best value for the target equations of 0.3959 and 0.4493, respectively, and can be summarized as presented in Table 22.

Table 22 shows that the work schedule for 20 nurses at 1500 rounds, when compared to the Lingo program, could calculatetheworkschedulewiththetargetequation(obj.)less than the genetic algorithm by around 5.80% and 10.8 ± 2.4%, on average, but that the one created by the Lingo program used longer computational runtime, by over 868.49% and 764.52%, on average. We can also notice that the results at 400 rounds reported the same direction; the best and the average solution of genetic algorithm differ about 9.82% and 14.97 ± 3.8% from the optimal, respectively. However, the optimal method used longer time, by over 3062.57% and

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 21: Results of genetic program calculated ten times at 400 rounds and 1500 rounds.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Target equation (obj.) number | Number of 20 nurses  Calculating cycle 400 | Calculating cycle 1500 | Number of 24 nurses  Calculating cycle 400 | Calculating cycle 1500 | | 1 | 0.4701 | 0.4578 | 0.5049 | 0.4730 | | 2 | 0.4361 | 0.4436 | 0.5406 | 0.4552 | | 3 | 0.4566 | 0.4772 | 0.4789 | 0.4493 | | 4 | 0.4812 | 0.4303 | 0.4590 | 0.5160 | | 5 | 0.4496 | 0.4781 | 0.4942 | 0.4586 | | 6 | 0.4920 | 0.4549 | 0.5383 | 0.4833 | | 7 | 0.4779 | 0.3959 | 0.5228 | 0.4626 | | 8 | 0.5428 | 0.4352 | 0.5133 | 0.4580 | | 9 | 0.5472 | 0.4368 | 0.5497 | 0.4985 | | 10 | 0.5231 | 0.4496 | 0.4551 | 0.4542 | | Mean | **0.4876** | **0.4459** | **0.5056** | **0.4708** | | Standard deviation | **0.038** | **0.024** | **0.033** | **0.021** |   Table 22: Comparison table for target equation (obj.) and duration in calculation of GA and Lingo program.   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | GA  Number of Number of nurses calculating  (person) cycles Target equation (obj.) | | Lingo %  Different in  Target Different result  Time (hour) Time (hour) computational equation (obj.) of obj.  run time | | | | | Lowest  1500  Mean 20 | 0.3959 0.4459 ± 0.024 | 6.03  6.85 | 0.3379 52.37 | 5.80%  10.80 ± 2.4% | 868.49%  764.53% | | Lowest 400 | 0.4361 | 1.71 |  | 9.82% | 3,062.57% | | Mean | 0.4876 ± 0.038 | 1.66 |  | 14.97 ± 3.8% | 3,154.82% |   Table 23: Comparison of differences in number of nurses, monthly expenses, and SD of work schedule created and actual work schedule in January 2012.   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  | Number of nurses |  | Monthly expense |  |  | SD |  | | Old | New Increase | Old | New | Decrease | Old | New | Decrease | | 18 | 20 2 | 395,964 | 366,400 | 29,564 | 1,089.212 | 214.843 | 80.28% |   Table 24: Comparison of differences in number of nurses, monthly expenses, and SD of work schedule created and actual work schedule in January 2013.  Number of nurses Monthly expense SD  Old New Decrease Old New Decrease Old New Decrease  23 20 3 416,548 366,400 50,148 1,966.735 214.843 89.08% |

3154.81%, on average, showing that the genetic program is more suitable for practical use.

# Conclusion

The findings showed that the genetic program created could manage the work schedule of nurses under given conditions and achieve the desired objective. It could create work schedules of nurses that were fair in overtime payment to all the nurses and could reduce the hospital expenses by setting the new coming nurses’ minimum salary at 15,000 Baht per person.

Table 25: Target equation and calculation time from genetic program.

Target equation (obj.)

Calculation time (hour)

0.3959

6.03

From Tables 23, 24, and 25, the following results can be concluded: the work schedules from proposed method reduce the waiting time of the number of patients waiting longer than 1.25 times of the average length nursing service from 34.08% to less than 15%, on average (starting from when the patients register and wait to see the doctor). If we want to schedule the work of the nurses in January 2012 in order to meet the criteria for the service of this study, we have to increase the number of nurses from 18 to 20 so that the number can perfectly meet the needs of the patients accordingtothecriteria.Theworkschedulesofthenursescan be most appropriate if the target equation is 0.3959 which uses the time of calculation as 6.03 hours with a standard deviation oflessthan80.28%andtheovertimepaymentlessthan29,564 Baht per month. When this is compared with the actual work schedule in January 2013, the work schedule could reduce the number of nurses from 23 to 20 with a decreased standard deviation, that is, 89.08%, and the decrease in the total overtime payment would be 50,148 Baht per month. We could calculate the most appropriate work schedule quicker than by using the conventional approach which results obtained by Lingo program, version 5.0, 868.49% with 1500 rounds for the calculation cycle.

As we know, hospitals belong to the service business sector; continuous improvement along with response to customer satisfaction is the key success factor in this business. To reduce the waiting time in every hospital process, high standards in medical care and highly experienced medical doctors are needed. Presently, Thailand has set its aim to become an Asian medical and tourist hub; this strategy of the country may cause and force hospitals to improve their service quality for responding well to the high number of outpatients. Updated data collection needs to be done; servicelevel policies have to be reconsidered. Demands of outpatient nurses and their schedules may change from time to time. Inpatient nurses may be allocated to assist when the management wishes to reduce the fluctuation in the number of outpatient nurses. Thus, modern management and high efficient tools are necessary; this is the reason why GA is involved instead of an adoption of the conventional approach. However, regarding the implementation of simulation, even GA is not familiar to the hospital people. Optimization capacitybuildingisanotherissuethatthehospitalmanagementwill have to consider.

# Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

# Acknowledgment

Chiang Mai University (CMU) through the research administration office provided budget to our Excellence Center in Logistics and Supply Chain Management (E-LSCM).

# References

1. T. Mongkolrangsarit and W. Thamapornpilat, *Development of Nurse Scheduling Method with Uncertain Workforce Demand*, Department of Industrial Engineering, Faculty of Engineering, Chulalongkorn University, 2007.
2. C. M. Rothe and H. B. Wolfe, “Cyclical scheduling and allocation of nursing staff,” *Socio-Economic Planning Sciences*, vol. 7, no. 5, pp. 471–487, 1973.
3. R. B. Norby, L. E. Freund, and B. Wagner, “A nurse staffing system based upon assignment difficulty,” *Journal of Nursing Administration*, vol. 7, no. 9, pp. 2–24, 1977.
4. T. Ryan, B. L. Barker, and F. A. Marciante, “A system for determining appropriate nurse staffing,” *Journal of Nursing Administration*, vol. 5, no. 5, pp. 30–38, 1975.
5. H. Wolfe and J. P. Young, “Staffing the nursing unit, Part I: controlled variable staffing,” *Nursing Research*, vol. 14, pp. 236– 243, 1965.
6. H. Wolfe and J. P. Young, “Staffing the nursing unit. II. The multiple assignment technique,” *Nursing Research*, vol. 14, no. 4, pp. 299–303, 1965.
7. J. L. Arthur and A. Ravindran, “ A multiple objective nurse scheduling model,” *AIIE Transactions*, vol. 13, no. 1, pp. 55–60, 1981.
8. I. Ozkarahan, “A disaggregation model of a flexible nurse scheduling support system,” *Socio-Economic Planning Sciences*, vol. 25, no. 1, pp. 9–26, 1991.
9. I. Ozkarahan and J. E. Bailey, “Goal programming model subsystem of a flexible nurse scheduling support system,” *IIE Transactions*, vol. 20, no. 3, pp. 306–316, 1988.
10. K. A. Dowsland, “Nurse scheduling with tabu search and strategic oscillation,” *European Journal of Operational Research*, vol. 106, no. 2-3, pp. 393–407, 1998.
11. P. Tormos, A. Lova, F. Barber, L. Ingolotti, M. Abril, and M. A. Salido, “A genetic algorithm for railway scheduling problems,” in *Metaheuristics for Scheduling in Industrial and Manufacturing Applications*, vol. 128 of *Studies in Computational Intelligence*, pp. 255–276, Springer, Berlin, Germany, 2008.
12. H. Kawanaka, K. Yamamoto, T. Yoshikawa, T. Shinogi, and S. Tsuruoka, “Genetic algorithm with the constraints for nurse scheduling problem,” in *Proceedings of the Congress on Evolutionary Computation*, vol. 2, pp. 1123–1130, Seoul, Republic of Korea, May 2001.
13. H. H. Millar and K. Mona, “Cyclic and non-cyclic scheduling of 12 h shift nurses by network programming,” *European Journal of Operational Research*, vol. 104, no. 3, pp. 582–592, 1998.
14. I. Blochlige, “Modeling staff scheduling problems. A tutorial,”¨ *European Journal of Operational Research*, vol. 158, no. 3, pp. 533–542, 2004.
15. B. Jaumard, F. Semet, and T. Vovor, “A generalized linear programming model for nurse scheduling,” *European Journal of Operational Research*, vol. 107, no. 1, pp. 1–18, 1998.
16. T. C¸ezik, O. Gunl¨ uk, and H. Luss, “An integer programming¨ model for the weekly tour scheduling problem,” *Naval Research Logistics*, vol. 48, no. 7, pp. 607–624, 2001.
17. Z. Wang, J.-L. Liu, and X. Yu, “Self-fertilization based genetic algorithm for university timetabling problem,” in *Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC ’09)*, pp. 1001–1004, ACM, Shanghai, China, June 2009.
18. J. Wang and K. Chu, “An application of genetic algorithms for the flexible job-shop scheduling problem,” *International Journal of Advancements in Computing Technology*, vol. 4, no. 3, pp. 271– 278, 2012.
19. M. Moz and M. V. Pato, “A genetic algorithm approach to a nurse rerostering problem,” *Computers & Operations Research*, vol. 34, no. 3, pp. 667–691, 2007.
20. C. Wotthipong, K. Soradech, and S. Nidapan, “The suitable genetic operators for solving the university course timetabling problem,” *Journal of Convergence Information Technology*, vol. 8, pp. 60–65, 2013.
21. K. Naveen, Karambir, and K. Rajiv, “A comparative analysis of PMX, CX and OX crossover operators for solving travelling salesman problem,” *International Journal of Latest Research in Science and Technology*, vol. 1, no. 2, pp. 98–101, 2012.

|  |  |  |
| --- | --- | --- |
|  |  |  |



**Details of the algorithms/approach**

Genetic algorithm (GA) is one of the well-known techniques from the area of evolutionary computation that plays a significant role in obtaining meaningful solutions to complex problems with large search space. GAs involves three fundamental operations after creating an initial population, namely selection, crossover, and mutation. Furthermore, GA is a powerful search and optimization algorithm, which is computational model based on Darwin's biological evolution theory of genetic selection and natural elimination. The GA, however, takes a long computation time in some specific problems because of its iteratively adaptive process for evolution. Therefore, it is indispensable to improve GA for reducing the computation time and preventing local minima efficiently.