A Simple Evolutionary Algorithm with Self-Adaptation for Multi-Objective Nurse Scheduling

Dario Landa-Silva[1] and Khoi N. Le[2]

[1] jds@cs.nott.ac.uk
[2] kxl@cs.nott.ac.uk
School of Computer Science
The University of Nottingham, UK

Summary. We present a multi-objective approach to tackle a real-world nurse scheduling problem using an evolutionary algorithm. The aim is to generate a few good quality non-dominated schedules so that the decision-maker can select the most appropriate one. Our approach is designed around the premise of 'satisfying individual nurse preferences' which is of practical significance in our problem. We use four objectives to measure the quality of schedules in a way that is meaningful to the decision-maker. One objective represents staff satisfaction and is set as a target. The other three objectives, which are subject to optimisation, represent work regulations and workforce demand. Our algorithm incorporates a self-adaptive decoder to handle hard constraints and a re-generation strategy to encourage production of new genetic material. Our results show that our multi-objective approach produces good quality schedules that satisfy most of the nurses' preferences and comply with work regulations and workforce demand. The contribution of this paper is in presenting a multi-objective evolutionary algorithm to nurse scheduling in which increasing overall nurses' satisfaction is built into the self-adaptive solution method.

1 Introduction

Producing good quality nurse schedules helps to provide better healthcare service, to improve overall job satisfaction and to make more efficient use of workforce. We are interested in tackling the nurse scheduling problem in a multi-objective fashion using an evolutionary algorithm. According to Ernst et al., the tendency in the modern workplace is to focus on individuals rather than on teams and hence, personnel schedules should cater to individual preferences [1]. This is particularly true in nurse scheduling because it is common that each nurse indicates his/her preference schedule. In our multi-objective approach, we set a target for nurse preference satisfaction and attempt to minimise the violation of soft constraints related to work regulations and

workforce demand. We refer to nurse scheduling as the construction of rosters for a ward of nurses over a short scheduling period (typically a few weeks). A roster can be defined as an assignment of personnel to specific shifts and/or duties. Here, a *nurse schedule* is a roster in which a line of work, made of shifts and days off, is assigned to each nurse in the ward over the scheduling period. For a discussion of other phases in the overall personnel scheduling process (e.g. demand modelling, task assignment, etc.) see [2]. Many healthcare institutions use some kind of software to aid the construction of nurse schedules but in many other cases this is still done manually [3]. For problems of considerable size, the non-automated construction of nurse schedules is time consuming, difficult and prone to errors. As Burke et al. note, "the automatic generation of high quality nurse schedules can lead to improvements in hospital resource efficiency, staff and patient safety, staff and patient satisfaction and administrative workload" [3].

Research into automated nurse scheduling has been very active in the last three decades or so. Methods applied to nurse scheduling include mathematical programming, goal programming, constraint programming, knowledge based systems, heuristics and meta-heuristics including evolutionary algorithms. Cheang et al. provide a brief literature review on the main models for nurse scheduling including types of constraints [4]. Burke et al. give a more comprehensive survey of the literature on automated nurse scheduling and classify papers with respect to nurse scheduling models and solution approaches [3]. Ernst et at. present surveys considering a wide range of personnel scheduling problems [1, 2].

Nurse scheduling problems are typically over constrained and tackling them with exact optimisation methods is difficult because considering all constraints leads to complex models. Therefore, approximation algorithms have been used in many of the nurse scheduling papers in the literature. In particular, heuristics and meta-heuristics have been very popular in recent years [3]. Reasons for this are that these methods can deal with the great number of existing constraints, they can be adapted to a wide range of problems and no mathematical models are required for their implementation. Nurse scheduling is a multi-criteria problem in which typically, work regulations, workforce demand, staff preferences and efficiency of service are in some kind of conflict. Most of the research on automated nurse scheduling has been conducted in the single-objective case using an aggregating penalty function to assess the quality of schedules. Several goal programming approaches in which criteria are prioritised and targets are set for each criterion, have been reported in the literature (e.g. [5, 6]). Not many applications of modern multi-objective meta-heuristics to nurse scheduling can be found in the literature (see surveys on the topic [1, 3, 4, 7]). One of the few examples is the Pareto simulated annealing algorithm for nurse scheduling in Polish hospitals [8]. That algorithm is a population-based method in which neighbourhood exploration is carried out as in the classical simulated annealing, but the search is guided using a weighted function in order to approach the trade-off surface [9].

In this paper, we apply a multi-objective evolutionary algorithm to tackle the problem of constructing schedules for a ward of nurses in the ophthalmological unit of the QMC hospital in Nottingham, UK. Our algorithm incorporates a re-generation strategy and a self-adative schedule decoder. The re-generation strategy replaces dominated solutions with new offspring in order to maintain diversity and re-activate the generation of high-quality solutions when the evolutionary process stagnates. The decoder is self-adaptive because it incorporates a self-mutation operator that adapts itself to the decoding process in order to repair hard constraint violations. Section 2 describes the QMC nurse scheduling problem and outlines previous work on this problem. Section 3 gives details of the solution encoding and its relation to the nurse's preference schedule. Section 4 describes our multi-objective approach in which nurse's preferences play a central role. Section 5 gives details of the self-adaptive schedule decoder incorporated in our algorithm. Section 6 presents experiments and results while final remarks are given in Section 7.

2 The QMC Nurse Scheduling Problem

2.1 Problem Description

The problem is to construct schedules for a ward of nurses in the Queens Medical Centre (QMC) in Nottingham, UK. The scheduling period is 28 days long. A ward typically consists of 20 to 30 nurses. Cover is required on a 24 hour basis, 7 days a week. Each nurse works either on a part-time or a full-time basis. Nurses are classified in a hierarchy according to their qualifications. Some nurses receive special training according to their ward. There are three types of shift: early, late and night. The early shift is from 07:00 to 14:45 counting for seven and a half hours (7.5 hours). The late shift is from 13:00 to 21:15 counting for seven and a half hours (7.5 hours). The night shift is from 21:00 to 07:15 counting for ten hours (10 hours). Occasionally, nurses indicate in their preference schedules the starting and finishing time that they prefer to work instead of one of the above 'normal shifts'. In that case, the 'unusual shift' is considered as the 'normal shift' (early, late or night shift) that covers most of the hours of the 'unusual shift'. For example, an 'unusual shift' from 09:00 to 17:00 is considered as an early shift. If the 'unusual shift' is equally spread over two adjacent 'normal shifts', one of these 'normal shifts' is uniformly chosen at random. For example, an 'unusual shift' from 17:00 to 01:15 can be considered as a late or as a night shift. The coverage demand, i.e. the required number of nurses with specific qualifications and training, is different for each shift. Nurses specify their individual working preferences (e.g. days off, preferred shifts, etc.) for each scheduling period. A number of working regulations (including nurses' annual leave) must be satisfied. Then, the problem is to construct a schedule that meets the workforce demand, satisfies all regulations and meets as many individual preferences as possible.

The QMC nurse scheduling problem includes the most common constraints in nurse scheduling literature as identified in [4]. We formulate this problem as the ordered pair:

$$NRP = \langle Nurses, C \rangle$$

where $Nurses = \{N_i : 1 \leq i \leq n\}$ is a set of n nurses and C is a set of constraints. Constraints in C can be hard (must be satisfied) or soft (should be satisfied). A nurse N_i is defined as follows:

$N_i = \langle NurseDetail_i, NursePreference_i, NurseSchedule_i, GeneSequence_i \rangle$

$NurseDetail_i = \langle Contract_i, Qualification_i, Trained_i, Hours_i \rangle$

$Contract_i \in \{FullTime, PartTime\}$ nurse N_i is full-time or part-time.

$Qualification_i \in \{RN, EN, AN, SN\}$ nurse N_i belongs to one of four qualification categories: registered (RN), enrolled (EN), auxiliary (AN) and student (SN). RNs and ENs are classified as qualified (QN) while QNs and ANs are both employed (PN). Qualified nurses, QNs, can receive additional training specific to the ward that they work in.

$Trained_i \in \{NoTrained, Trained\}$ in the ophthalmological ward, a nurse can receive eye-training.

$Hours_i \in \mathbf{N}^+$ is the number of contracted hours for nurse N_i, for full-time nurses $Hours_i$ is 75 hours per fortnight, for part-time nurses $Hours_i$ is per week and as specified in their individual contract.

$NursePreference_i = \{p_{i,j} : 1 \leq j \leq NoOfDays\}$ is the nurse's preference schedule for the scheduling period, where $NoOfDays$ is the length of the scheduling period, 28 in the QMC problem, and $p_{i,j} \in \{AnnualLeave, Any, DayOff, Early, Late, Night\}$ is the nurse's preference for day j, Any indicates no specific preference.

$NurseSchedule_i = \{s_{i,j} : 1 \leq j \leq NoOfDays\}$ is an individual nurse's schedule, i.e. a string containing the assigned shift for each day in the scheduling period, where $s_{i,j} \in \{AnnualLeave, DayOff, Early, Late, Night\}$. A ward schedule for the QMC problem is a collection of n individual nurse schedules.

$GeneSequence_i = Permutation\{shift : 1 \leq shift \leq NoOfShifts\}$ is the gene representation used in the evolutionary algorithm implemented in this paper, where $NoOfShifts = NoOfDays * 3$ i.e. the total number of shifts in the scheduling period. This representation is illustrated in detail in Section 3.

2.2 Hard Constraints

OneShiftADay A nurse works at most one shift (*Late*, *Early*, *Night*) per day.

MaxHours Nurses can work a maximum number of hours (given by $Hours_i$) over a period of time according to their individual contract.

MaxDaysOn The maximum number of consecutive days that a nurse can work, which is 6. This constraint guarantees regular breaks for nurses.

MinDaysOn The minimum number of consecutive days that a nurse can work. This value is normally 2 for full time nurses. It is not applicable for most part time nurses because of the fewer number of shifts that they work.

Succession Defines illegal shift combinations for nurses. A *Night* shift must not be followed by an *Early* shift.

HardRequest Defines nurses' requests that must be satisfied. For example, *annual leave* requests in the preference schedule are considered hard requests.

2.3 Soft Constraints

SoftRequest Defines nurses' requests that are desirable but might be violated. In the QMC problem, these requests are typically for working on specific shifts (*Early*, *Late*, *Night*, *DayOff* and 'unusual shifts').

SingleNight A nurse should not be assigned an individual *Night* shift. Nurses at the QMC ward prefer to work night shifts in blocks of two or more. This applies to all full time nurses and certain types of part time nurses whose individual contracts are at least 20 hours a week.

WeekendSplit Nurses prefer to work both days of the weekend or none at all.

WeekendBalance The maximum number of weekends that nurses may work over the scheduling period. In the QMC ward, nurses may not work more than 3 out of 4 consecutive weekends.

Coverage A certain number of nurses with specific qualifications and specific training should be assigned to particular shifts as shown in Table 1.
It should be noted that it is not necessary to assign 6 different nurses to the *Early* shift to meet the *Coverage* requirements. This demand can be satisfied with only 4 nurses if all of them are *qualified*, one of them is *registered* and one of them has received *eye-training*.

CoverageBalance The number of nurses assigned to each shift over the scheduling period should be evenly distributed. Any surplus/shortage of nurses over the scheduling period should be kept to a minimum. This constraint prevents an excessive number of nurses being assigned to a particular shift while having a shortage of nurses in other shifts.

Table 1. *Coverage* demand of nurses in each shift.

	Early	Late	Night
QNs	4	3	2
RNs	1	1	0
ETs	1	1	1

Table 2. Measurement of *CoverageBalance*.

QNs	Early	Late	Night
Demand	4	3	2
Assigned	4	4	1
Difference	0	1	-1

All hard constraints must be satisfied for a schedule to be feasible. We assess the quality of a feasible schedule by measuring the violation of the six soft constraints but always taking into account the preferences expressed by each nurse. To measure the satisfaction of soft constraints (with the exception

of *CoverageBalance*), we simply count the number of violations of each soft constraint type. However, the violation of a soft constraint is not penalised if the shifts assigned to the nurse's schedule comply with the nurse's preferences expressed in $NursePreference_i$. More precisely, we measure the violation of soft constraints as follows.[3]

SoftRequest$(p_{i,j}, s_{i,j})$ if the assigned shift $s_{i,j}$ is not as the nurse's preferred shift $p_{i,j}$, a penalty of 1 is applied. No penalty is applied if a working shift $p_{i,j}$ (*Early*, *Late*, or *Night*) is requested and a *DayOff* $s_{i,j}$ is assigned.

SingleNight(N_i, D) if a *Night* shift is assigned to nurse N_i on day D, and shifts different to *Night* are assigned on adjacent days ($D-1$ and $D+1$), and the assigned shifts are not as in $NursePreference_i$, a penalty of 1 is applied.

WeekendSplit(N_i, D) if nurse N_i is assigned to work only on one of days D or $D+1$ of a weekend, and the assigned shifts are not as in $NursePreference_i$, a penalty of 1 is applied.

WeekendBalance(N_i) if nurse N_i is assigned to work at least one day in each of the four weekends in the scheduling period, and the assigned shifts are not as in $NursePreference_i$, a penalty of 1 is applied.

Coverage(*shift*) if the number of nurses with specific qualifications and training assigned to a given shift is less than the coverage demand, a penalty equal to the deficit in the number of nurses assigned is applied.

CoverageBalance we measure the satisfaction of this constraint using statistical variation on the difference between the number of qualified nurses assigned to each shift and the coverage demand for qualified nurses. For example, for a schedule of 1-day and 3 shifts (*Early*, *Late*, *Night*), the difference between coverage demand and assigned nurses is calculated as in Table 2. Then, *CoverageBalance* is measured as the variation on the *Difference* set of $3*NoOfDays$ shifts. In this example, the penalty of the *CoverageBalance* constraint has a value of $\frac{2}{3}$ (0.667).

We split the six soft constraints into four groups and each group corresponds to an objective function. Group 1 consists of the *SoftRequest* constraint measuring the level of nurse preferences satisfaction. Group 2 consists of *SingleNight*, *WeekendBalance*, and *WeekendSplit* constraints measuring satisfaction of work regulations. Group 3 consists of the *Coverage* constraint measuring shortfalls in workforce demand. Group 4 consists of the *CoverageBalance* constraint measuring the distribution of nurses in the schedule to ensure a balanced coverage for the whole scheduling period. In the QMC problem, nurses satisfaction is at the centre of the scheduling process due to the shortfall of staff in hospitals recently. Therefore, when constructing a schedule with our multi-objective approach, we pre-set a target value for objective 1 to guarantee a minimum level of staff satisfaction. Moreover, nurses' preferences are

[3] Full description of the QMC nurse scheduling problem, examples of how the violation of soft constraints is measured and data sets are available at the following url http://www.cs.nott.ac.uk/∼kxl/research/QMC/qmc.html

also taken into account when dealing with the other three objectives which are subject to optimisation (this is explained in detail in the following sections). The aim is to produce a set of schedules in low computational time, all with the required level of staff satisfaction but representing a set of compromise alternatives with respect to the other three objectives. Then, the decision-maker (typically a senior nurse) can select the most appropriate schedule for the given planning period.

2.4 Previous Work on the QMC Problem

The QMC nurse scheduling problem was also tackled by Beddoe and Petrovic using a combination of case-based reasoning and tabu search concepts [10, 11]. Our results are not easily comparable to those obtained by Beddoe and Petrovic because of two reasons. One is that they tackled the QMC problem in a single-objective manner attempting to find one feasible solution, while we seek a set of alternative schedules. The other reason is that they tackled a simplified version of the problem. In their study, Beddoe and Petrovic considered constraints *OneShiftADay*, *MaxHours*, *MaxDaysOn*, *MinDaysOn Succession*, *Coverage*, *HardRequest* and *SoftRequest* only. However, we applied our approach to the data sets used by Beddoe and Petrovic and results are reported in Section 6.

3 Schedule Encoding and Construction

We represent a solution (ward schedule) to the QMC nurse scheduling problem as a set of n sub-solutions. Each sub-solution is a list of *NoOfShifts* shifts corresponding to an individual nurse's schedule. In many nurse scheduling papers, the approach is to start from an empty schedule. Instead, we take the set of preference schedules into account. Nurses indicate their preferred shifts in the preference schedule ($NursePreference_i$) while the constructed schedule ($NurseSchedule_i$) indicates the actual shifts assigned. Figure 1 illustrates a preference and a constructed schedule for one nurse on a 7-day scheduling period where E, L, N, O correspond to *Early*, *Late*, *Night*, and *DayOff* shifts respectively. An empty cell in the preference schedule indicates no preference for that day. In the constructed schedule, an empty cell represents a *DayOff*. In this example, the nurse has to work in days 1 to 5 and only one of the preferred shifts (for day 2) was satisfied. We use an indirect representation in our evolutionary algorithm in which a permutation list of integers from 1 to $3 * NoOfDays$ is decoded to create an individual nurse schedule. Figure 1 also illustrates a permutation list for a 7-day scheduling period.

Starting from the left, the decoder reads a shift x from the permutation list ($1 \leq x \leq 3 * NoOfDays$) and decodes it to the corresponding day and shift (*Early*, *Late*, *Night* correspond to 0, 1, 2 respectively) in the constructed schedule as follows: day $D = (x - 1) \operatorname{div} 3 + 1$, shift $S = (x - 1) \bmod 3$. For

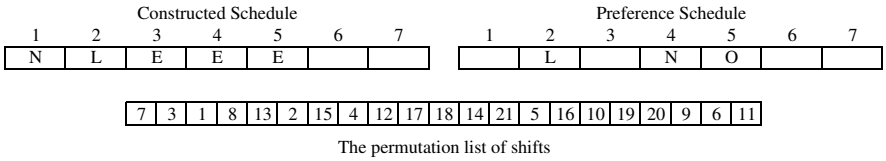


Fig. 1. Constructed Schedule, Preference Schedule and Permutation List

example, $x = 7$ in the permutation list represents an *Early* shift ($S = 0$) in day 3 ($D = 3$). When assigning shifts, the decoder ensures the satisfaction of all hard constraints (see Section 5). The decoder stops assigning shifts to the nurse's schedule when the end of the permutation list is reached or the total number of working hours is within a threshold τ given by:

$$MaxHours - MinShiftLength < \tau \leq MaxHours$$

where *MaxHours* is as defined in Section 2 and *MinShiftLength* is the length of the shortest shift (7.5 hours in the QMC problem).

4 The Proposed SEAMO-R Algorithm

SEAMO, the Simple Evolutionary Algorithm for Multi-Objective Optimisation was proposed by Valenzuela who also showed that this algorithm outperforms other state-of-the-art Pareto-based evolutionary algorithms on the multiple knapsack problem [12] with respect to *the size of space covered* $\mathcal{S}$ and *the coverage of two sets* $\mathcal{C}$ indicators ($\mathcal{S}$ and $\mathcal{C}$ are defined in [13]). SEAMO uses a steady-state population and a simple elitist replacement strategy. The algorithm chooses each member of the population, in turn, to be the first parent and a second parent is chosen at random. Offspring is produced by applying cycle crossover [14] on the two parents followed by a single mutation. If the offspring's objective vector improves on any *best-so-far* objective function, it replaces one of the parents and the objective's *best-so-far* is updated. Otherwise, if the offspring dominates one of the parents, it replaces that parent (unless it is a duplicate, then the offspring is deleted). SEAMO2, an updated version of SEAMO was presented later and it was observed that a more elaborate replacement strategy improved the performance of the algorithm on both combinatorial and continuous multi-objective problems [15]. SEAMO2 is different from SEAMO in that SEAMO2 allows the offspring to replace an individual in the population that it dominates if the offspring and both parents are mutually non-dominated while in SEAMO the offspring was discarded in this case.

We implemented both versions of SEAMO on the QMC problem and observed that a major drawback was that no good offspring was generated after only few generations. Therefore, we designed a re-generation strategy that

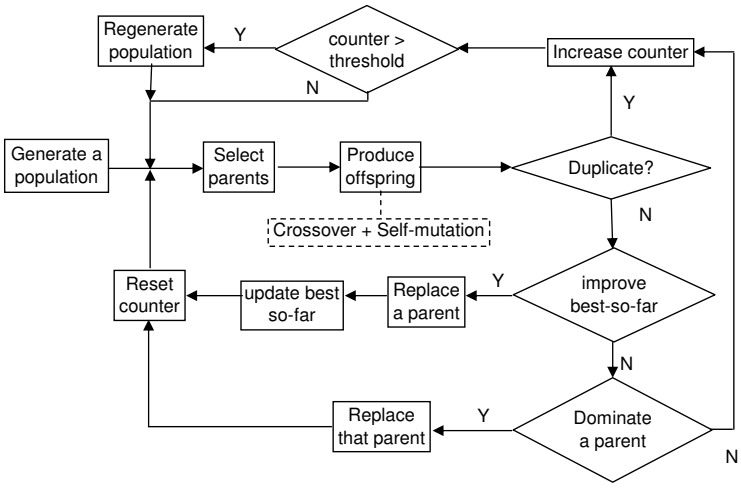


Fig. 2. The SEAMO-R algorithm

activates the production of high-quality offspring to tackle this stagnation issue. In this paper, we exploit the SEAMO's concept and propose SEAMO-R (simple evolutionary algorithm for multi-objective with re-generation), a variation of the SEAMO approach, to tackle the QMC nurse scheduling problem. Figure 2 illustrates the SEAMO-R algorithm.

The population *re-generation strategy* (Regenerate population in Figure 2) re-activates the generation of high-quality offspring when the evolutionary process stagnates. Improving the population in SEAMO-R relies entirely on the replacement strategy and the re-generation strategy. The purpose of the re-generation strategy is to maintain diversity in the population and to produce good offspring. This strategy replaces a portion of dominated solutions with new solutions. The *Regeneration-Probability* parameter controls the rate of replacing a dominated solution with a new solution. A probability of 1 means that all dominated solutions will be replaced. A probability of 0 means that no dominated solutions will be replaced and the re-generation mechanism is deactivated. This replacement process is triggered if after a number of evaluations given by *Regeneration-Rate*, there is no replacement of solutions in the population with offspring. The *Regeneration-Rate* parameter is important on the performance of SEAMO-R. If the *Regeneration Rate* is too high, the population will be frequently re-generated and the population hardly evolves. If the *Regeneration-Rate* is too low (or zero as in SEAMO), the evolutionary process falls into stagnation and is very difficult to produce offspring to replace solutions in the population. The *Regeneration-Rate* is highly dependant on the *Population-Size* and the *Regeneration-Probability* (we describe later how we set these parameters in our experiments). Full details of SEAMO-R are given in the pseudocode below:

```
Procedure SEAMO-R
Begin
   Regeneration-Rate = ⟨pre-defined by user⟩
   Regeneration-Probability = ⟨pre-defined by user⟩
   Population-Size = P
   non-improve-counter = 0
   Soft-Request-Probability = ⟨pre-defined by user⟩
   Generate a random population of P schedules
   Evaluate the objective vector for each schedule and store it
   Record the best-so-far for each objective function
   Repeat
      For each schedule in the population (1st parent)
          Select a second schedule at random (2nd parent)
          Apply crossover to produce offspring
          Decode permutation list for each nurse's schedule in the offspring
          Evaluate the objective vector for the offspring
          If offspring's objective vector improves on any best-so-far
             Then the offspring replaces one of the parents and best-so-far is updated
                     non-improve-counter = 0
             Else If the offspring is not duplicate and dominates 1st parent (or 2nd parent)
                     Then the offspring replace that parent
                             non-improve-counter = 0
                     Else non-improve-counter = non-improve-counter + 1
                  EndIf
          EndIf
          If non-improve-counter ≥ Regeneration-Rate (**Re-generation Starts Here**)
             Then replace all dominated schedules in the current population
                             with probability of Regeneration-Probability
                             by schedules generated uniformly at random
                  non-improve-counter = 0
          EndIf
      Endfor
   Until stopping condition satisfied
   Print all non-dominated solutions in the final population
End
```

5 Decoder and the Hard Constraints

5.1 Self-adaptive Schedule Decoder

The performance of SEAMO-R on the QMC nurse scheduling problem is significantly better than the performance of SEAMO and SEAMO2 because of our re-generation strategy. The performance of SEAMO-R on the QMC nurse scheduling problem is further improved by using a self-adaptive decoder to handle hard constraints. SEAMO-R also incorporates a self-mutation operator that works according to the current state of the decoding process. The decoder chooses, from either the preference schedule or the offspring's genetic material, a shift *S'* to assign to day *D*. The self-mutation operator swaps the shift *S'* with the left-most shift in the offspring's gene sequence which is associated to day *D*. If the shift *S'* chosen from the preference schedule is a *DayOff*, there is no self-mutation and the decoder moves to the next shift in the gene sequence. During the decoding process, the *Soft-Request-Probability* indicates the rate at which a shift in the preference schedule is used by the decoder rather than the current shift in the offspring's genetic material. A probability of 0 means that the decoder uses the offspring's genetic material without considering

the preference schedule. A probability of 1 means that the decoder uses the preferred shift first and if this shift is not suitable, the decoder then uses the offspring's genetic material. Details of the self-adaptive decoder are shown in the following pseudocode:

```
Procedure QMC-Decoder
Begin
  Repeat
    Select a nurse schedule at random
    Assign all hard requests to that nurse's constructed schedule
    For each shift S in the permutation list of this nurse
      If the nurse schedule is fully assigned (depended on nurse's MaxHours constraint)
        Then terminate the decoding process (terminate the For loop)
      EndIf
      If day D associated with this shift S is not yet assigned
        Then
          If assign(S, D) violates Succession constraint
            Then choose shift S' from Preference Schedule
                 with probability Soft-Request-Probability or from permutation list
                 (assign(S', D) does not violates Succession)
              If assign(S', D) does not violate MaxDaysOn, MaxHours, HardRequets
                 and the Coverage demand for shift S' of day D is not exceeded
                 Then assign S' to D, and apply the self-mutation process on S'
              EndIf
            Else
              If assign(S, D) violates MinDaysOn constraint
                Then choose shift S' from Preference Schedule with
                     probability Soft-Request-Probability or from permutation list
                     to assign to day D' adjacent to day D
                     (assign(S, D) and assign(S', D') does not violates Succession)
                  If assign(S, D) and assign(S', D') do not violate
                     MaxDaysOn, MaxHours, HardRequets and Coverage demands
                     for shift S of day D and shift S' of day D' is not exceeded
                     Then assign S to D, S' to D'
                          and apply the self-mutation process on S'
                  EndIf
                Else
                  If the Coverage demand for shift S of day D is not exceeded
                    Then assign S to D
                  EndIf
              EndIf
          EndIf
      EndIf
    EndFor
    Evaluate objective functions
  Until all nurse schedules are decoded
End
```

The adaptive strategy in our decoder aims to guide the construction of schedules taking into account all nurses' preferences. There is no guarantee that a given permutation, once decoded, will always produce the same schedule. This is because our decoder incorporates stochastic elements that help to explore different possibilities. In general, there are two ways to satisfy hard constraints when constructing a nurse schedule. One is to only accept the assignment of shifts that maintain feasibility. The other is to repair an infeasible schedule by changing the shift assigned to day D or changing the shift assigned to an adjacent day. In this paper, we use the first approach to deal with the *OneShiftADay*, *MaxHours*, *MaxDaysOn* and *HardRequest* hard constraints. We use the second approach to deal with the *Succession* and *MinDaysOn* hard constraints. We repair a violation of the *Succession* hard constraint by

changing the shift assigned to day D such that a *Night* shift is not followed by an *Early* shift. We repair a violation of the *MinDaysOn* hard constraint by assigning a working shift to an adjacent day. We take care to ensure that the *Succession* hard constraint is not violated while repairing the *MinDaysOn* hard constraint.

A number of self-adaptive approaches have been proposed in the context of evolutionary algorithms. Most approaches focus mainly on adjusting parameter settings (such as the probability of mutation and recombination) and selecting the evolutionary operators (from a range of operators available) to be applied at different times during the search. For example, Meyer-Nieberg and Beyer proposed a self-adaptive *punctuated crossover* that adapts the number and location of crossover points [16] while Sereni et al. proposed a self-adaptive recombination approach in which the crossover operator is chosen at random [17]. For reviews on adaptation and self-adaption in evolutionary algorithms see [16, 18, 19, 20]. Note that our self-adaptive mutation operator adjusts its characteristics throughout the search process by adapting the rate of mutation, the position of mutation points and the number of mutation points. That is, our mutation strategy 'learns' the gene sequence of individuals in order to identify good genes and appropriate positions for those good genes in the gene sequence. The mutation strategy does this by shifting genes that violate hard constraints to the end of the gene sequence, and shifting genes that provoke less violations to the beginning of the gene sequence. As the search progresses, the mutation strategy gradually shifts 'promising genes' to the beginning of the gene sequence leading to a quicker decoding process and subsequently a reduction in the number of mutation points and the mutation rate. In the rest of this Section we give more details on how the self-adaptive mutation operator works.

5.2 Handling *Succession*

We illustrate how the decoder deals with the *Succession* hard constraint. The decoder assigns shifts to the constructed schedule as in Figure 3. The decoder reads shifts in the permutation list and assigns each shift to the corresponding day unless that day is already occupied or the *Succession* hard constraint is violated. Note that in Figure 3, decoding 4 from the permutation list provokes a violation of the *Succession* constraint (a *Night* shift is followed by an *Early* shift). The decoder repairs this violation using the self-mutation process. The shift to repair this violation can be chosen from the preference schedule or from the permutation list according to the *Soft-Request-Probability*.

Figure 4 illustrates the self-mutation process to repair a violation of the *Succession* hard constraint using a shift from the preference schedule. After assigning *Early* shift to day 3, *Night* shift to day 1, and *Early* shift to day 5, the decoder skips shift 2 and shift 15 in the permutation list because day 1 and day 5 in the constructed schedule are already occupied. Shift 4 is read from the permutation list. However, assigning shift 4 (*Early* shift to day 2) violates

the *Succession* constraint. A high *Soft-Request-Probability* forces the decoder to use the preference schedule and then a *Late* shift is assigned to day 2. This corresponds to assigning shift 5 instead of shift 4 from the permutation list. Therefore, the self-mutation process is applied to the current permutation list to swap shift 4 and shift 5. In this case, assigning the shift in the preference schedule does not violate the *Succession* constraint. However, if assigning the shift in the preference schedule violates the *Succession* constraint, the decoder uses the permutation list as an alternative to find a suitable shift (though the decoder was forced to consider the preference schedule first). If the decoder uses a shift *DayOff* from the preference schedule, no self-mutation is required and the decoder continues with the next shift in the permutation list.

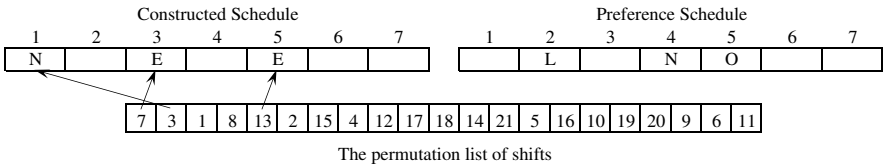


Fig. 3. The decoding process

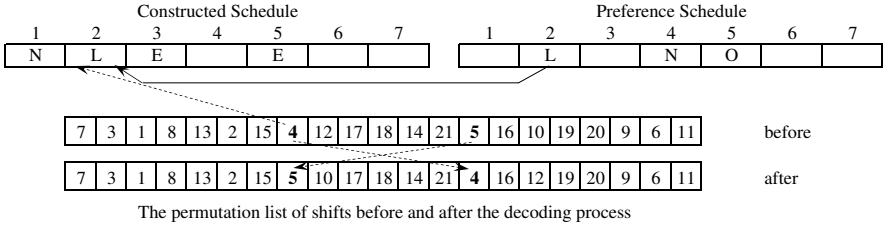


Fig. 4. Repairing the violation of the *Succession* constraint using self-mutation

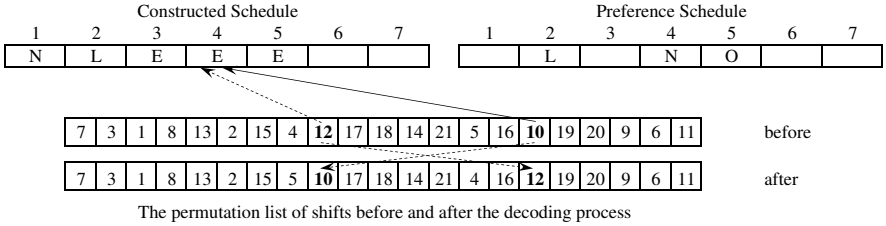


Fig. 5. Repairing the violation of the *Succession* constraint using self-mutation

Figure 5 illustrates the self-mutation process to repair a violation of the *Succession* hard constraint using a shift from the permutation list. The decoder attempts to assign shift 12 from the permutation list (*Night* shift to day 4). However, this violates the *Succession* constraint by creating a

Night-Early shift combination in day 4 and day 5. A low value of *Soft-Request-Probability* forces the decoder to use the permutation list. The decoder searches for the first shift in the permutation list (reading from left to right) that can be assigned to day 4 without violating the *Succession* constraint. In this case, shift 10 in the permutation list (*Early* shift in day 4) is selected. The decoder assigns shift 10 and self-mutation is applied to swap shift 12 and shift 10 in the permutation list.

5.3 Handling *MinDaysOn*

To handle the *MinDaysOn* hard constraint, the decoder works on the same principle as for the *Succession* constraint. The difference is that to repair violations of the *Succession* constraint, the decoder searches for a suitable shift *S'* to assign to day *D* and the shifts assigned to adjacent days are already known. However, to repair violations of the *MinDaysOn* constraint, the decoder searches for a shift *S'* to assign to day *D'* which is adjacent to day *D* and the shift *S* assigned to day *D* is already known. First, the decoder identifies a list of suitable shifts which do not create a shift combination that violates the *Succession* constraint. These shifts are associated with either day *D-1* or day *D+1*. From this list, the shift that appears first in the permutation list from left to right, is assigned to its associated day. The self-mutation process is then triggered. The repair of the *MinDaysOn* hard constraint is illustrated in Figure 6. The decoder continues moving to the right of the permutation list (Figure 7). Shift 12 is then assigned to the constructed schedule causing a violation of the *MinDaysOn* constraint. Therefore, the decoder has to search for an additional shift to repair the violation. The *Night* shift is assigned to day 3 in the constructed schedule (shift 9 in the permutation list). The self-mutation process swaps shift 9 with the left most shift in the permutation list associated with day 3 (shift 7 in the permutation list). In this case, we still assume that the *Soft-Request-Probability* instructs the decoder to select the additional shift from the permutation list.

Now, assume that the decoder is instructed to select an additional shift from the preference schedule when repairing violations. In Figure 8 the decoder moves right on the permutation list and assigns shift 18 (*Night* shift to day 6). An additional shift is required to repair the violation of the *MinDaysOn* constraint. The decoder searches in the preference schedule for suitable shifts. The only suitable shift in the preference schedule is the *Late* shift in day 7 (shift 20 in the permutation list) because assigning the *Early* shift to day 5 creates a violation of the *Succession* constraint (*Night-Early* shift combination in day 4 and day 5). The decoder assigns shift 20 to the constructed schedule and the self-mutation swaps shift 20 and shift 19 in the permutation list. If the decoder cannot find any suitable shift in the preference schedule to repair the violation, the decoder will look at the permutation list to find a suitable shift. For example, if the preferred shift in day 7 is the *Early* shift, there is no suitable shift in the preference schedule. Then, the decoder searches in the

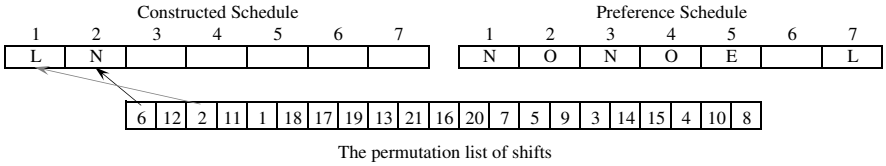


Fig. 6. Repairing the violation of the *MinDaysOn* constraint using self-mutation

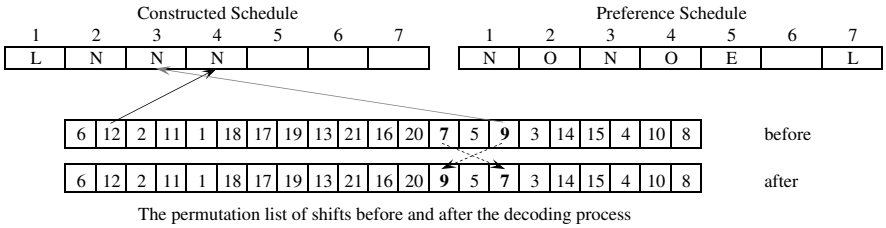


Fig. 7. Repairing the violation of the *MinDaysOn* constraint using self-mutation

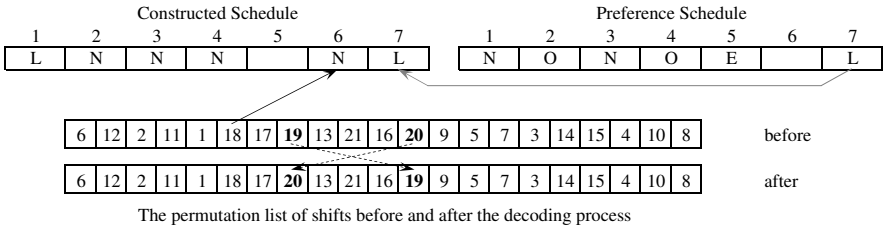


Fig. 8. Repairing the violation of the *MinDaysOn* constraint using self-mutation

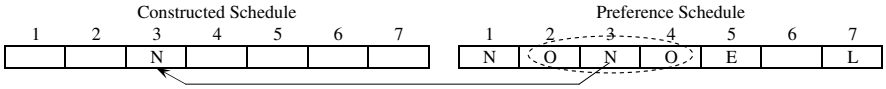


Fig. 9. Repairing the violation of the *MinDaysOn* constraint using self-mutation

permutation list and finds shift 21. The self-mutation then swaps shift 21 and shift 19 in the permutation list.

Note that in the preference schedule there is a combination *DayOff-Night-DayOff* from day 2 to day 4 (Figure 9). If at any point the decoder attempts to assign *Night* shift to day 3 while day 2 and day 4 are free, then assigning *Night* shift to day 3 is not considered a violation of the *MinDaysOn* constraint. This is because the *DayOff-Night-DayOff* shift combination from day 2 to day 4 is the one indicated in the preference schedule. That is, violations of the *MinDaysOn* and *Succession* constraint are permitted if the nurse's preferences indicate so.

Besides repairing the *Succession* and *MinDaysOn* hard constraints, the decoder also attempts to minimise the number of surplus nurses in each shift. This is to reduce the number of violations of the *Coverage* constraint and equally distribute nurses amongst all shifts. This is achieved by only assigning shifts to days if the coverage demand has not been exceeded yet. Therefore, the decoder assigns shifts to days if and only if the above condition is met and the *MaxHours*, *MaxDaysOn* and *HardRequest* hard constraints are not violated. Satisfaction of *OneShiftADay* is ensured by the encoding scheme. As indicated in Section 3, the decoder assigns shifts until the end of the permutation list is reached or the total number of working hours is within the threshold τ.

6 Experiments and Results

6.1 Experimental Setting

There are 7 data sets for the QMC nurse scheduling problem, one for each scheduling period from March to September 2001. Our first experiment was to identify appropriate parameter settings for SEAMO-R. We set *RegenerationProbability* = 0.75 and *SoftRequestProbability* = 0.60 using data from March2001 as a training set. The *SoftRequestProbability* is set according to the required level of nurses' preferences satisfaction. We performed 30 independent runs. Each run took between 4 and 5 minutes on a 2.2GHz AMD Opteron x86 64-bit Processor with Linux O/S (SuSe 9.0). We used values of *RegenerationRate* set to of 100, 200, 500 and 750. The *PopulationSize* was set to values of 100, 200 and 500 with the *NumberOfIterations* set to 15000, 7500 and 3000 generations respectively. Our preliminary results suggested that SEAMO-R performs the best when using a *PopulationSize* = 200, *NumberOfIterations* = 7500 and *RegenerationRate* = 500.

6.2 Performance of SEAMO-R

We carried out 30 runs with the above parameter settings for each of the other 6 data sets (April2001 to September2001). We found around 7 non-dominated schedules in each run. We calculate the similarity between two ward schedules as follows. Consider shift $s^1_{i,j}$ in schedule 1 and shift $s^2_{i,j}$ in schedule 2, where $1 \leq i \leq n$ (n is the number of nurses), $1 \leq j \leq NoOfDays$. Then:

$$s^1_{i,j}, s^2_{i,j} \in \{O, E, L, N\} \text{ (where } O \text{ is } DayOff\text{)}$$

$$matched_{1,2} = \left|\left\{(i,j) \mid s^1_{i,j} = s^2_{i,j} \wedge (s^1_{i,j} \neq O \vee s^1_{i,j} \neq O)\right\}\right|$$

$$unmatched_{1,2} = \left|\left\{(i,j) \mid s^1_{i,j} \neq s^2_{i,j} \wedge (s^1_{i,j} \neq O \vee s^1_{i,j} \neq O)\right\}\right|$$

$$Similarity_{1,2} = \frac{matched_{1,2}}{matched_{1,2} + unmatched_{1,2}}$$

The average similarity for the whole set of non-dominated solutions in the final population is calculated as the mean of all similarities between each pair of non-dominated solutions.

Table 3. Average results produced by SEAMO-R for the QMC problem

Period	ND	Obj2	Obj3	Obj4	Obj1	AverSimil
March2001	7.367	4.149	6.363	0.274	89.9%	78.4%
April2001	7.700	3.633	10.282	0.276	89.1%	79.3%
May2001	7.033	2.924	17.779	0.249	90.3%	85.2%
June2001	3.933	1.164	40.619	0.259	92.9%	91.5%
July2001	7.900	3.977	41.202	0.300	87.4%	80.1%
August2001	7.200	4.171	10.277	0.246	90.2%	81.8%
September2001	7.467	3.357	21.755	0.267	89.3%	84.0%

To illustrate the overall quality of the schedules generated with our approach, we show in Table 3 the 'average results' for each data set. *ND* is the average number of non-dominated solutions in the final population over the 30 independent runs. *Obj1* is the level of nurses' preferences satisfaction and is measured as a percentage as follows:

$$Obj1 = \frac{\text{Total number of requests - } \textit{SoftRequest} \text{ violation}}{\text{Total number of requests}}$$

Obj2 is the total number of violations of the *SingleNight*, *WeekendSplit* and *WeekendBalance* constraints. *Obj3* is the number of violations of the *Coverage* constraint. *Obj4* is the number of violations of the *CoverageBalance* constraint. *AverSimil* is the average similarity over the 30 independent runs for each data set. Remember that *Obj1* is set as target while the other three objectives are subject to optimisation. We computed these results using the set of all non-dominated solutions obtained in the 30 runs for each data set. Details of all constructed non-dominated schedules are available from the authors. We note that given the highly constrained nature of nurse scheduling problems, it is often very difficult to find feasible schedules. This was the case in the QMC problem too and hence the need for the self-adaptive decoder and the re-generation strategy. Therefore, it was not surprising that relatively few non-dominated solutions were obtained by the end of each run. However, this number of schedules is adequate because it would be very difficult for a senior nurse to select a ward schedule from a larger set of alternatives. It is also important to note that the similarity between non-dominated schedules is high because our approach seeks to match the nurse preference schedules according to the *Soft-Request-Probability* set by the user.

Table 3 shows that the average nurses' preference satisfaction is approximately 90% for all 7 data sets. In our results, the preference satisfaction of each non-dominated solution in the final population is in the range $90\% \pm 3\%$. One can easily realise that the values for *Obj3* (violations of the *Coverage* constraint) are quite different amongst the 7 data sets. This value is quite low

for March2001, April2000 and August2001 whereas for June2001 and July2001 it is noticeably high. A close examination of the data sets reveals that this is because the number of available staff-hours is quite different amongst the 7 instances. We estimate the number of available staff-hours for each of the data sets as follows: March2001 (2200), April2001 (2100), May2001 (1950), June2001 (1700), July2001 (1700), August2001 (2100) and September2001 (1930). Taking into account the coverage demand of *qualified nurses (QNs)* (see Table 1), it can be estimated that there must be at least 2030 staff-hours available to fulfill this demand. However, it is difficult to fulfill this requirement with exact 2030 staff-hours due to the existence of other constraints and individual requests. We estimate that there should be about an extra 150 staff-hours in order to minimise the number of violations. For those months with a shortage in available staff-hours (e.g. June2001 and July2001), such shortage affects mainly the provision of qualified nurses to the *Night* shift and this contributes to violations of the *Coverage* constraint. This is because with a limited number of available staff-hours, the coverage demand in shifts *Early* and *Late* is satisfied first as these shifts count for 7.5 hours. However, the *Night* shift counts for 10 hours and satisfying the coverage demand in this shift required extra staff-hours. That explains why with about 400 extra staff-hours, the number of violations of the *Coverage* constraint (*Obj3*) in March2001 is about 35 less than the number of violations in June2001 or July2001. Note that the number of violations in *Obj2* (total violations of *SingleNight*, *WeekendSplit* and *WeekendBalance*) is very low, only between 2 and 3 violations within a full schedule.

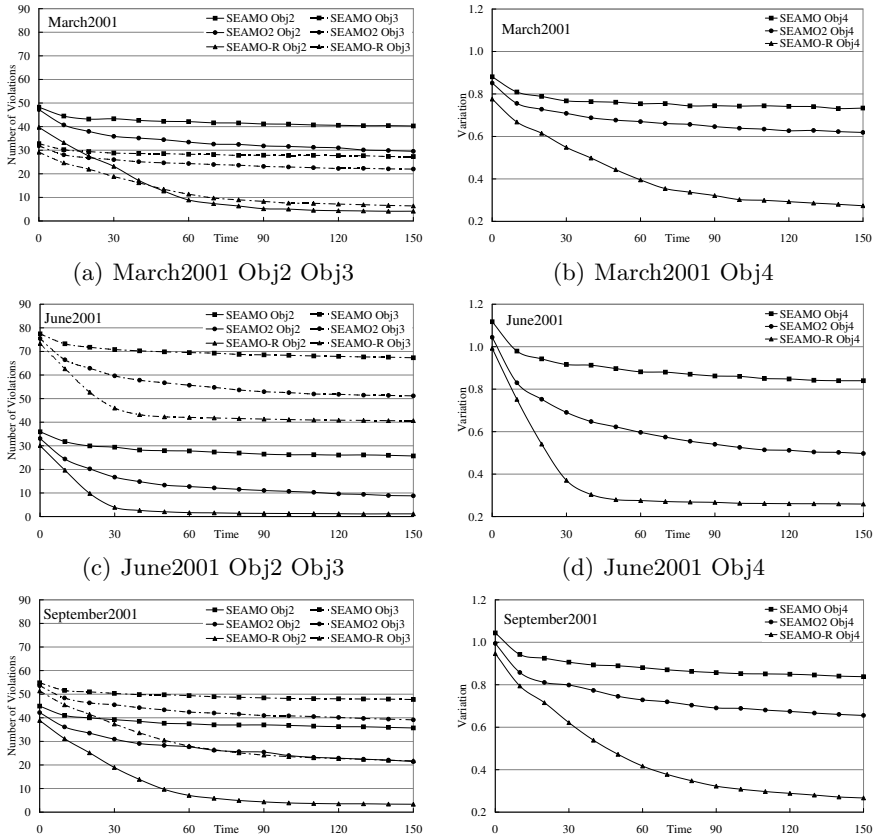
6.3 Comparison with SEAMO and SEAMO2

Our next set of experiments was to compare SEAMO-R against SEAMO and SEAMO2 on the QMC problem in order to assess the contribution of our regeneration strategy and self-adaptive decoder on the good results obtained. We incorporated the self-adaptive decoder into SEAMO and SEAMO2 for these experiments. Average results are presented in Table 4 and it is clear that SEAMO-R outperforms SEAMO and SEAMO2.

Table 4. Performance of SEAMO, SEAMO2 and SEAMO-R on the QMC problem

Period	SEAMO			SEAMO2			SEAMO-R		
	Obj2	Obj3	Obj4	Obj2	Obj3	Obj4	Obj2	Obj3	Obj4
Mar2001	40.274	27.320	0.734	29.522	22.001	0.619	4.149	6.363	0.274
Apr2001	40.301	33.858	0.778	29.045	28.018	0.666	3.633	10.282	0.276
May2001	32.471	44.083	0.831	22.076	36.695	0.672	2.924	17.779	0.249
Jun2001	25.691	67.312	0.840	8.791	51.145	0.497	1.164	40.619	0.259
Jul2001	27.824	63.776	0.844	15.963	55.121	0.637	3.977	41.202	0.300
Aug2001	38.590	34.849	0.779	26.796	28.996	0.658	4.171	10.277	0.246
Sep2001	35.686	47.760	0.838	21.423	39.126	0.656	3.357	21.755	0.267

We also examined the performance of SEAMO-R, SEAMO and SEAMO2 throughout the search process. We traced the evolution of the average values for the set of non-dominated solutions at every 50 generations in each of the 30 runs and for all 7 data sets. In Figure 10, we only present graphs for 3 data sets, March2001, June2001, September2001 which are representative of all our results. The graphs show that SEAMO-R quickly outperforms SEAMO and SEAMO2 and overall, SEAMO-R improves the quality of the non-dominated solutions very rapidly. While the replacement strategy in SEAMO2 helped this algorithm to outperform SEAMO on multi-objective benchmark problems [15], our experiments show that our re-generation strategy contributes substantially to the good results obtained on the highly constrained QMC nurse scheduling problem.



(a) March2001 Obj2 Obj3 (b) March2001 Obj4

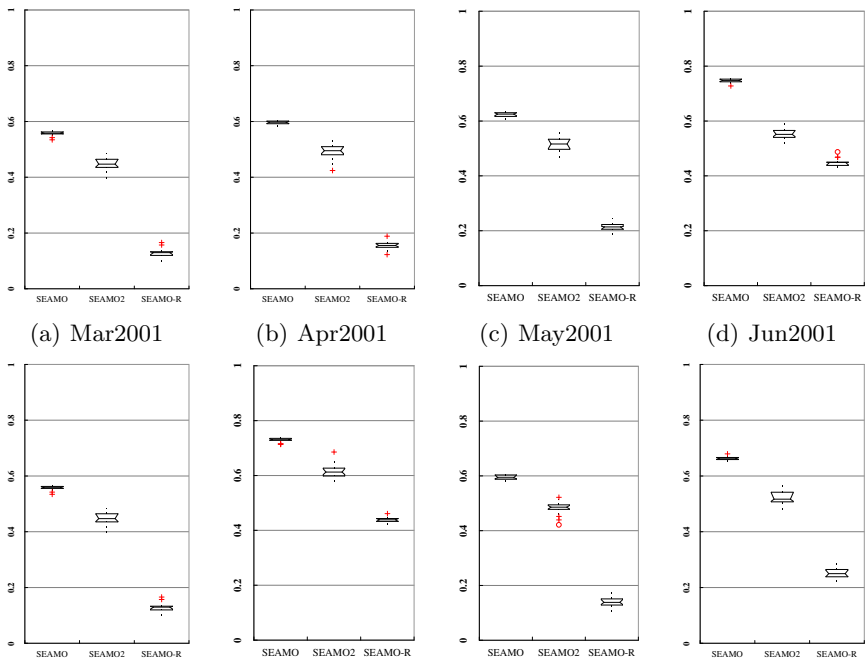
(c) June2001 Obj2 Obj3 (d) June2001 Obj4

(e) September2001 Obj2 Obj3 (f) September2001 Obj4

Fig. 10. Performance of SEAMO-R, SEAMO and SEAMO2 on the QMC problem

Regarding multi-objective optimisation, we evaluate the Pareto fronts produced by SEAMO, SEAMO2, SEAMO-R using two metrics, *size of the space*

covered $\mathcal{S}$ and *coverage of two sets* $\mathcal{C}$, proposed in [13]. The $\mathcal{S}$ hypervolume metric is scaled as the percentage of the volume created by the origin and the reference point (100, 100, 3) with respect to (*Obj2*, *Obj3*, *Obj4*). The reference point is estimated using the average objective vector's value of the non-dominated solutions in the initial population. With respect to the *coverage* metric $\mathcal{C}$, all non-dominated solutions in the final population of SEAMO-R dominate the ones of SEAMO and SEAMO2 for all 7 data sets. Figure 11 measure the percentage of non-dominated objective space. The horizontal axes present SEAMO, SEAMO2, SEAMO-R. The *size of the space covered* produced by SEAMO-R is much better than the one of SEAMO and SEAMO2 for all 7 data sets. As it can be seen, the results for June2001 and July2001 are not as good as March2001 because of the shortage of available staff-hours which was explained above.



(a) Mar2001 (b) Apr2001 (c) May2001 (d) Jun2001

(e) Mar2001 (f) Jul2001 (g) Aug2001 (h) Sep2001

Fig. 11. Performance of SEAMO-R, SEAMO and SEAMO2 on the QMC problem based on *size of the uncovered space* $\mathcal{S}$

6.4 Selecting a Ward Schedule

Although we recorded all non-dominated solutions in the final population for each run, here we simulate the decision-making process of choosing one

schedule from the set of alternatives. We assume that a 'best schedule' is chosen based on the following priority:

1. number of violations of *SingleNight*, *WeekendSplit*, *WeekendBalance* (*Obj2*)
2. number of violations of *Coverage* (*Obj3*)
3. the penalty for *CoverageBalance* (*Obj4*)
4. the overall nurses' preferences satisfaction *SoftRequest* (*Obj1*)

However, different decision makers could use different priorities and a different schedule from the obtained non-dominated set would be chosen. We present the objective values for these 'best schedules' in Table 5.

Table 5. A selected 'best schedule' for each data set

Period	Obj2	Obj3	Obj4	Obj1
March2001	2.567	5.8	0.271	89.7%
April2001	2.000	9.767	0.275	88.9%
May2001	2.267	16.5	0.231	90.1%
June2001	0.867	39.977	0.261	92.8%
July2001	1.900	39.900	0.318	87.2%
August2001	3.467	8.233	0.217	90.0%
September2001	2.567	20.167	0.253	89.2%

6.5 Previous Results on the QMC Problem

As it was mentioned in Subsection 2.4, Beddoe and Petrovic used a simplified version of the QMC problem and tackled it in a single-objective manner [10, 11]. We applied our SEAMO-R approach to the data sets (March2001 and April2001) used by Beddoe and Petrovic and results are reported in Table 6. We can see that the results obtained by SEAMO-R are slightly worse than those reported by Beddoe and Petrovic. However, note that the number of violations of the *Coverage* constraint (*Obj3*) produced by SEAMO-R on the Beddoe and Petrovic data sets is only slightly better than the number of violations on the data sets of this paper (Table 3), although the later data sets correspond to much more constrained instances. Full details of the comparison with the work of Beddoe and Petrovic are available in [21]. In order to facilitate further research and comparison with our results, we make the QMC problem instances available in the web page mentioned in Section 2.

Table 6. Average results of SEAMO-R on Beddoe and Petrovic data sets [10, 11]

	SEAMO-R		CABAROST(CB-OBJ-TL-R10	
Period	Obj3	Obj1	Obj3	Obj1
March2001	2.600	88.6%	0.100	90.1%
April2001	4.900	89.3%	0.000	90.7%

7 Final Remarks

We have presented a multi-objective evolutionary approach to tackle a real-world nurse scheduling problem in which the satisfaction of staff preferences drives the search for non-dominated solutions. We described an adaptation of the Simple Evolutionary Algorithm for Multi-objective Optimisation (SEAMO) to tackle a nurse scheduling problem with four objectives. One of the objectives is set as a target while the other three are subject to optimisation. In our multi-objective approach, we have grouped soft constraints in a manner that is meaningful to the decision-maker (usually a senior nurse). The target objective is associated to the satisfaction of nurses' preferences. The other three objectives are associated to 1) meeting work regulations, 2) meeting coverage demand and, 3) ensuring balanced coverage demand for the whole scheduling period. We developed a re-generation strategy to aid diversification and a self-adaptive decoder to repair constraint violations. These two mechanisms are driven by the target level of nurses' preferences satisfaction which can be set by the decision-maker. The resulting algorithm is SEAMO-R, a Simple Evolutionary Algorithm with Re-generation for Multi-objective Optimisation. The re-generation strategy replaces dominated solutions with new ones to avoid stagnation. The self-adaptive decoder uses the nurse preference schedule, a random permutation of shifts and a self-mutation operator to construct schedules and maintain feasibility. Our results show that SEAMO-R produces sets of good quality of feasible and non-dominated ward schedules for the QMC nurse scheduling problem.

References

1. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B., Sier, D.: An annotated bibliography of personnel scheduling and rostering. Annals of Operations Research **127** (2004) 21–144
2. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: a review of applications, methods and models. European Journal of Operational Research **153** (2004) 3–27
3. Burke, E.K., De Causmaecker, P., Vanden Berghe, G.: The state of the art of nurse scheduling. Journal of Scheduling **7** (2004) 441–499
4. Cheang, B., Li, H., Lim, A., Rodrigues, B.: Nurse rostering problems: a bibliographic survey. European Journal of Operational Research **151** (2003) 447–460
5. Authur, J.F., Ravindran, A.: A multiple objective nurse scheduling model. AIIE Transactions **13**(1) (1981) 55–60
6. Berrada, I., Ferland, J.A., Michelon, P.: A multi-objective approach to nurse scheduling with both hard and soft constraints. Socio-Economic Planning Sciences **30**(3) (1996) 183–193
7. Landa Silva, J.D., Burke, E.K., Petrovic, S.: An introduction to multiobjective metaheuristics for scheduling and timetabling. In: Gandibleux X., Sevaux M., Sorensen K., T'kindt V. eds., Metaheuristic for multiobjective optimisation, Lecture Notes in Economics and Mathematical Systems **535** (2004) 91–129

8. Jaszkiewicz, A.: A metaheuristic approach to multiple objective nurse scheduling. Foundations of Computing and Decision Sciences **22**(3) (1997) 169–183
9. Czyzak, P., Jaszkiewicz, A.: Pareto simulated annealing - a metaheuristic for multiple-objective combinatorial optimization. Journal of Multicriteria Decision Analysis **7**(1) (1998) 34–47
10. Beddoe, G.R., Petrovic, S.: Combining case-based reasoning with tabu search for personnel rostering problems. Computer Science Technical Report No. NOTTCS-TR-2004-5, The University of Nottingham (2004)
11. Beddoe, G.R., Petrovic, S.: Enhancing case-based reasoning for personnel rostering with selected tabu search concepts. To Appear in The Journal of The Operational Research Society (2007)
12. Valenzuela, C.L.: A simple evolutionary algorithm for multi-objective optimization (seamo). IEEE World Congress on Computational Intelligence (WCCI2002): Congress on Evolutionary Computation (CEC2002), IEEE press (2002) 717–722
13. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation **3**(4) (1999) 257–271
14. Oliver, I.M., Smith, D.J., Holland, J.R.C.: A study of permutation crossover operators on the traveling salesman problem. Genetic Algorithms and their Application: Proceedings of the Second International Conference on Genetic Algorithms (1987) 224–230
15. Mumford, C.L.: Simple population replacement strategies for a steady-state multi-objective evolutionary algorithm. 2004 Genetic and Evolutionary Computation Conference (GECC0 2004), LNCS, Springer **3102** (2004) 1389–1400
16. Meyer-Nieberg, S., Beyer, H.G.: Self-adaptation in evolutionary algorithms. In Lobo, F.G., Lima, C.F., Michalewicz, Z., eds.: Parameter Setting in Evolutionary Algorithms. Volume 54. Springer (2007) 47–76
17. Sareni, B., Regnier, J., Roboam, X.: Recombination and self-adaptation in multi-objective genetic algorithms. In: Artificial Evolution, LNCS. Volume 2936. Springer (2004) 115–126
18. Hinterding, R., Michalewicz, Z., Eiben, A.E.: Adaptation in evolutionary computation: a survey. In: 1997 IEEE International Conference on Evolutionary Computation. (1997) 65–69
19. Bäck, T.: Self-adaptation in genetic algorithms. In Varela, F.J., Bourgine, P., eds.: Proceedings of the 1st European Conference on Artificial Life (ECAL 1992), MIT Press (1992) 227–235
20. Angeline, P.J.: Adaptive and aelf-adaptive evolutionary computations. In Palaniswami, M., Attikiouzel, Y., eds.: Computational Intelligence: A Dynamic Systems Perspective. IEEE Press (1995) 152–163
21. Le, K.N.: An evolutionary algorithm for multi-objective nurse scheduling. Master Thesis, School of Computer Science and IT, The University of Nottingham (2006)