Storing and Adapting Repair Experiences in Employee Rostering — Source link

Sanja Petrovic, Gareth Beddoe, Greet Van den Berghe

Institutions: University of Nottingham

Published on: 21 Aug 2002

Topics: Constraint (information theory)

Storing and Adapting Repair Experiences in Employee Rostering

Sanja Petrovic[1], Gareth Beddoe[1], and Greet Vanden Berghe[2]

[1] Automated Scheduling, Optimisation, and Planning Research Group,
School of Computer Science and Information Technology, University of Nottingham,
Jubilee Campus, Nottingham NG8 1BB, UK
`sxp,grb@cs.nott.ac.uk`
[2] KaHo St-Lieven, Information Technology,
Campus Rabot, Gebroeders Desmetstraat 1, B-9000 Gent, Belgium
`greetvb@kahosl.be`

Abstract. The production of effective workforce rosters is a common management problem. Rostering problems are highly constrained and require extensive experience to solve manually. The decisions made by expert rosterers are often subjective and are difficult to represent systematically. This paper presents a formal description of a new technique for capturing rostering experience using case-based reasoning methodology. Examples of previously encountered constraint violations and their corresponding repairs are used to solve new rostering problems. We apply the technique to real-world data from a UK hospital.

1 Introduction

The rostering of employees within an organisation must satisfy operational, legal and management requirements whilst taking into account the conflicting considerations of staff morale and sensible working practice. Manual rostering experts develop strategies for balancing these requirements, drawing on their extensive experience to make rostering decisions. This paper describes a method for capturing this experience for re-use in an automated setting.

Capturing such rostering knowledge in the form of logical rules (e.g. IF THEN rules) is difficult and can lead to incomplete and inflexible domain models [15]. This is because rostering decisions are made by often subjective interpretations of the subtle interactions of a number of parameters. We move away from explicit representations of rostering rules and introduce a system that stores them implicitly in a history of past experience.

Case-based reasoning (CBR) [11] is an artificial intelligence methodology that aims to imitate human style decision making by solving new problems using knowledge about the solutions to similar problems. CBR methodology operates under the premise that similar problems will require similar solutions. Previous problems and solutions are stored in a *case-base* and accessed during reasoning by processes of identification, retrieval, adaptation and storage. The identification and retrieval phases search the case-base for cases containing problems that are most similar to the current problem in terms of a set of characteristic features called *indices*. The solutions from these retrieved cases are then adapted and applied to the context of the current problem. If the new solution might be useful for solving future problems then it is stored as a new case in the case-base. CBR is suited to the problem of capturing rostering knowledge because it enables us to build a history of correspondences between constraint violations and their solutions.

A number of different approaches have been used for solving employee rostering problems in the past including linear and integer programming [3,4,14,18], goal programming [2,5], and constraint satisfaction techniques [1,8,12,13]. CBR was employed by Scott and Simpson [16] by storing shift patterns used for the construction of rosters. A number of meta-heuristic methods have also been developed with some success using tabu search [6,9,10] and memetic algorithms [7]. These methods traverse the search space through neighbourhoods defined using extensive domain knowledge and experimental trial and error.

The knowledge capturing technique described here aims to provide a means to intelligently and dynamically define neighbourhoods tailored to individual problems. The repairs used to solve constraint violations are stored in a case-base of previous experience and are adapted when new violations are encountered. This method is not intended to produce final solutions to rostering problems but instead works on the more *local* level by repairing individual constraint violations. It provides a means by which to store and re-use rostering experience and could in the future be incorporated in some form of metaheuristic or other problem solving framework.

We investigated the problem of rostering nurses in an ophthalmological ward at the Queens Medical Centre University Hospital Trust (QMC) in Nottingham, United Kingdom. Section 2 of this paper will describe the problem in this ward including the manual procedures used at present. Sections 3 and 4 will define the problem mathematically and present the CBR based local repair algorithms that have been developed. An example of a problem solving instance is given in Section 5. The results of some experiments are provided in Section 6 before a discussion on the future directions of this research.

2 Problem Description

The rostering problem at the QMC is rather more complex than many previously investigated in the literature in terms of the levels of detail that must be considered. The descriptions of nurses qualifications and abilities does not lend itself well to the problem subdivision methodologies of [2,10] – we cannot simply divide the nurses up into disjoint "levels" of qualification. This also results in more complex constraint descriptions in the form of appropriate skill mixes and cover requirements for the ward.

The problem consists of assigning shifts to nurses over a set time period (usually 28 days) subject to a number of constraints. Nurses have one of four levels of "qualification" depending on the training they had to become a nurse. These are, in descending order of seniority: Registered (RN), Enrolled (EN), Auxiliary (AN) and Student (SN). RNs are the most qualified and have had extensive training in both the practical and management aspects of nursing whereas ENs have had less training in only practical nursing. ANs are unqualified nurses who can perform basic duties and SNs are training to be either RNs or ENs. Three additional qualification types are used to group these "real" types. Registered and enrolled nurses are grouped together as Qualified Nurses (QN) and RNs, ENs, and ANs, are classified as Employed Nurses (PN). The classification XN groups all of the nurses together.

In addition to these qualifications nurses can receive specialty training specific to the ward they work on (in this ophthalmological ward it is "eye-training" (ET)). A grade is also assigned to each nurse and is determined by a combination of their qualification, specialty training and the amount of practical experience they have. These grades range from A to I with I being the most senior. Two final attributes taken into account during the rostering process are gender (M or F) and international status (I or H). The latter is important in UK hospitals due to an increased reliance on overseas-trained nurses to overcome staff shortages in the public sector. In this paper we refer to all of these attributes of a nurse as their "descriptive features".

At present, roster production in the QMC ward is a three-stage process. The self-rostering planning approach is used to collect shift preference information from all members of staff (see [17] for a comprehensive survey of the use of self-rostering in UK hospitals). This approach recognises that nurses are professionals who will fulfil their responsibilities without excessive administrative intervention.

The three stages of roster production are

1. nurses are assigned to teams (according to a particular skill mix);
2. nurses produce partial rosters (called preference rosters) for the planning period in consultation with other members of their teams;
3. partial rosters are combined to produce the ward roster which is invariably infeasible; any constraint violations are repaired by senior staff members.

These preference rosters indicate when individual nurses would like to work a particular shift, and when they would like a day off. If they have no preference then they can leave a particular day blank. The amount of detail and flexibility for shift assignments varies considerably between nurses.

We wish to automate the third stage of the process. At present, this stage takes a considerable number of hours per month to complete. The constraint violations present in the roster need to be repaired whilst retaining as much preference information as possible.

At this stage of this research we consider only a subset of the constraints that are present in the real-world problem. A number of hard and soft constraints can be identified but here we shall deal only with the two most important hard constraints. It is our conjecture that some of the soft constraints will be satisfied as a consequence of the information stored about the repair of hard constraint violations – or at least that a series of these repairs will minimise the degree of violation of the soft constraint.

The two hard constraints (hereinafter referred to simply as “constraints”) considered here are

- *Cover* constraints define the skill mix required for a particular shift. They are described by variables indicating the type and number of nurses needed for a specified shift. For example, the early shift requires 4 QNs;
- *Totals* constraints describe the maximum working hours allowed over a particular period. These can be defined for all nurses of a specific type as well as for individual nurses over any time period. For example, the maximum number of hours that can be (legally) worked by any nurse (XN) within a fortnight is 75.

Three basic rostering actions, or repair types, have been identified and these are representative of the kind of actions carried out by rostering experts. REASSIGN repairs are the simplest and involve reassigning a nurse's shift on a particular day. Two nurses can have their shifts on a particular day swapped by the SWAP repair. The final repair type, SWITCH, interchanges the shifts assigned to one nurse on two different (consecutive or non-consecutive) days. These basic repair types require different data and this will be reflected in the mathematical representation described in the following section.

The method proposed in this paper stores information about individual repairs of constraint violations in rosters. It keeps a history consisting of pairs of problems and solutions from which new repairs can be generated.

3 Mathematical Formulation

We define a rostering problem as an ordered pair

$$R = \langle N, C \rangle ,$$

where $N = \{nurse_i : 0 \leq i < n\}$ is the set of nurses to be rostered such that

$$nurse_i = \langle NurseType_i, hours_i, NR_i, NP_i \rangle .$$

$NurseType_i = \{f_{i,1} \ldots f_{i,I}\}$ is an array of descriptive features where

$$f_{i,1} \in \{RN, EN, AN, SN, QN, PN, XN\}$$

is the nurse's qualification and $f_{i,2} \ldots f_{i,I}$ describe gender, international status, specialty training, and grade. $hours_i \in \mathbb{R}^+$ is the number of hours the nurse is contracted to work in a week (normally 37.5).

$$NR_i = \{s_{i,j} : 0 \leq j < period\}$$

is a set of assignment variables $s_{i,j}$ which represent the actual shift assignment for $nurse_i$ on day j over the number of days for which the roster has been constructed, *period*.

$$NP_i = \{p_{i,j} : 0 \leq j < period\}$$

is a set of variables $p_{i,j}$ representing the preferred assignment of $nurse_i$ on day j. The variables $s_{i,j}$ and $p_{i,j}$ can take values from the set {UNASSIGNED, EARLY, LATE, NIGHT, OFF}.

The set C consists of a number of constraints that can take one of the following types:

$$COVERCONSTRAINT(NurseType, shift, minimumCover)\,;$$
$$TOTALSCONSTRAINT(NurseType, period, maximumHours)\,.$$

Cover constraints describe the minimum number *minimumCover* of nurses of type *NurseType* who must be assigned *shift* on every day of the roster. Totals constraints describe the maximum number of hours *maximumHours* that nurses of type *NurseType* may work over a number of days *period.*

When constraints are applied to N, they generate a set of violations of a type corresponding to the constraint type. We define the problem instance spaces P_v^R and P_r^R as the set of violations and possible repairs given the current roster R. An element $violation_\alpha \in P_v^R$ details the type of violation and the parameters relevant to it. It may be one of the following types:

$$COVER(NurseType, day, shift)\,;$$
$$TOTALS(nurse_i, startDay, endDay)\,.$$

Cover violations are generated by cover constraints and represent that the number of nurses of type *NurseType* assigned *shift* on the *day* indicated is insufficient. Likewise, totals violations are generated by totals constraints and describe that $nurse_i$ has been assigned too many hours between days *startDay* and *endDay.*

An element $repair_\beta \in P_r^R$ describes the type of repair and the nurses, days, and shift assignments involved. They can be one of the following types:

$$REASSIGN(nurse_i, day_\beta, shift_\beta)\,;$$
$$SWAP(nurse_i, nurse_j, day_\beta)\,;$$
$$SWITCH(nurse_i, day1_\beta, day2_\beta)\,.$$

Reassign repairs assign $shift_\beta$ to $nurse_i$ on day day_β. Swap repairs interchange the shift assignments of $nurse_i$ and $nurse_j$ on day day_β and switch repairs interchange the shift assignments of $nurse_i$ on the days $day1_\beta$ and $day2_\beta$.

The violation and repair problem spaces represent information relevant to a specific instance of a rostering problem (an instantiation of R). The nurses, days and shifts they describe refer only to those specified by R. In order to store and reuse examples of previous violation/repair experiences we need to define a generalised structure.

We define the case-base CB as a set of previously encountered violations and their corresponding repairs. The case-base $CB = W_v \times W_r$ where W_v is the space of stored violations (the problem history) and W_r is the space of stored repairs (the solution history). Therefore a case $c_\gamma = (v_\gamma, r_\gamma) \in CB$ where $\gamma \in \mathit{\Gamma}$. We define v_γ and r_γ as follows:

$$v_\gamma = \langle ViolationType_\gamma, ViolationIndices_\gamma \rangle \,,$$
$$r_\gamma = \langle RepairType_\gamma, RepairIndices_\gamma \rangle \,.$$

Here $ViolationType_\gamma$ and $RepairType_\gamma$ contain the type and necessary parameters describing generalised violations and repairs. $ViolationIndices_\gamma$ and $RepairIndices_\gamma$ store the feature information needed to match problem instances during case retrieval, and to generate repairs during case adaptation. The index sets are arrays of feature values, which are generally integers or real numbers, and in the case of the repair indices store shift pattern information. This index information will be described in more detail later in the paper.

We can now define the generalisation functions $\theta_v^R : P_v^R \to W_v$ and $\theta_r^R : P_r^R \to W_r$ that map instance specific violations and repairs respectively to their generalised case representations. The various types of violation and repair are converted to their generalised equivalents and the indices necessary for retrieval and adaptation are calculated. For example, the $TOTALS$ violation and its parameters will be mapped:

$$\theta_v^R\Big(TOTALS(nurse_i, startDay, endDay)\Big)$$
$$= \langle CBTOTALS(NurseType_i), ViolationIndices \rangle \,,$$

and similarly for a $SWAP$ repair:

$$\theta_r^R\Big(SWAP(nurse_i, nurse_j, day)\Big)$$
$$= \langle CBSWAP(NurseType_i, NurseType_j, s_{i,day}, s_{j,day}), RepairIndices \rangle \,.$$

The violation and repair indices used are described in Section 4.

Figure 1 gives a graphical summary of the generalisation functions. Note that here we define only the transformation from problem instance to case-base. The inverses of these functions are not well-defined mathematically and in fact would not make sense from an operational point of view. The method of generating new repairs using a combination of current problem information and historical experience is the subject of the next section.

In order to describe the retrieval and adaptation processes the definitions of equality of some variables need to be defined. Two nurses are considered to have the same type if all of their descriptive feature information is the same. We define equality of the $NurseType$ variable mathematically as follows. Given $NurseType_a = \{f_{a_1}, \ldots, f_{a_I}\}$ and $NurseType_b = \{f_{b_1}, \ldots, f_{b_I}\}$

$$NurseType_a = NurseType_b \text{ iff } f_{a_i} = f_{b_i} \forall i, (1 \leq i \leq I) \,.$$

Equality of the generalised $ViolationType$ and $RepairType$ information are defined similarly. This information is in the following form:

$$TYPENAME_a(param_{a_1}, \ldots, param_{a_M}) \,.$$

Then equality between variables $ViolationType_a$ and $ViolationType_b$, or $RepairType_a$ and $RepairType_b$ occurs if and only if

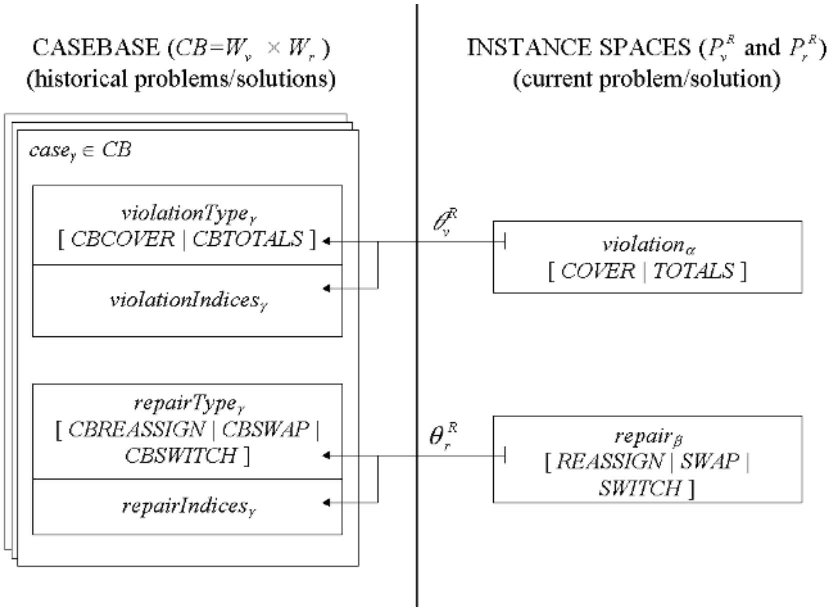
CASEBASE ($CB = W_v \times W_r$)
(historical problems/solutions)
INSTANCE SPACES (P_v^R and P_r^R)
(current problem/solution)
$case_y \in CB$
$violationType_y$
[CBCOVER | CBTOTALS]
$violationIndices_y$
$repairType_y$
[CBREASSIGN | CBSWAP | CBSWITCH]
$repairIndices_y$
θ_v^R
θ_r^R
$violation_\alpha$
[COVER | TOTALS]
$repair_\beta$
[REASSIGN | SWAP | SWITCH]

Fig. 1. Relationship between the current instance and the cases from the case-base

$$(TYPENAME_a = TYPENAME_b) \wedge (param_{a_i} = param_{b_i}) \forall i, (1 \le i \le M) .$$

It remains to define two functions that will be used in the following section. The similarity measure function Sim is a standard nearest-neighbour method common in the CBR literature [11]. We apply it here to the index sets of violations and repairs. Given $IndexSet_a = \{index_{a_1}, \ldots, index_{a_I}\}$ and $IndexSet_b = \{index_{b_1}, \ldots, index_{b_I}\}$, representing either $ViolationIndices$ or $RepairIndices$ sets,

$$\textsc{Sim}(IndexSet_a, IndexSet_b) = \left(\frac{1}{I} \sum_{i=1}^{I} w_i \times dist(index_{a_i}, index_{b_i}) \right)^{-1} ,$$

where I is the number of elements in the index sets, w_i are the index weights, and

$$dist(index_{a_i}, index_{b_i}) = \left| \frac{index_{b_i} - index_{a_i}}{index_{max_i} - index_{min_i}} \right| .$$

The values $index_{max_i}$ and $index_{min_i}$ are the maximum and minimum values for the corresponding index recorded in the case-base. The *dist* function therefore finds the normalised distance between feature values. By weighting each of these distances we assign a relative importance to each feature. In this research these weights are generally flat (all set to 1). The effect of changes to these values will be the subject of future investigation.

During repair adaptation it is necessary to compare the shift patterns of nurses in the current roster with stored patterns in the *RepairIndices* variables. The shift pattern comparison function *comp* compares the shift patterns of two different nurses over a five day period (two days before and after the day on which a repair is applied). The difference in shift pattern strings is calculated as follows. Given $nurse_\alpha$ with shift pattern string $x_1x_2x_3x_4x_5$ and a pattern $y_1y_2y_3y_4y_5$ to be compared we define

$$comp_\alpha(y_1y_2y_3y_4y_5) = \sum_{i=1}^{5} \delta_\alpha(y_i)\,,$$

where

$$\delta_\alpha(y_i) = \begin{cases} 0 \; y_i = x_i\,, \\ 1 \; y_i \neq x_i\,. \end{cases}$$

By this definition the shift patterns *EUUNN* and *EEUUL* have a difference of $0 + 1 + 0 + 1 + 1 = 3$.

4 Retrieval and Adaptation

The main difference between the classical OR and meta-heuristic approaches to nurse rostering described in the literature and the CBR approach proposed here is that the repairs generated are not optimal in any sense. We have described no measures of the quality of repairs – the aim here is to imitate, as closely as possible, the decisions made by rostering experts without relying on quantified measures of roster quality.

The notion of problem similarity is key to the success of any CBR application. A method must be developed for finding the most similar problems in the case-base to the current problem being solved. Having identified the most similar problems the stored repairs have to be adapted to the context of the current roster. Here again we use the notion of similarity. We must generate a new repair that most closely matches the repair stored in the case.

There are a number of justifications for taking this approach. The aim must not be to replicate exactly the repair from the retrieved case. This would be incorrect in most situations as the nurses and time periods involved would undoubtedly be different. In some instances it may even be impossible if a different set of nurses is defined. It is also clear that simply generating a random repair of a specified type would not be correct. A compromise between these two extreme approaches is reached between by the generation of repairs that are considered *similar* to the retrieved repairs. This emphasis on the similarity of repairs in different instances motivated the generalisations of both violation and repair described in the previous section.

The retrieval process is split into two distinct searches of the case-base. The first search filters the case-base to obtain cases containing violations that match the current problem in terms of violation type and parameters. This is a strict

Table 1. Violation indices

Global problem characteristics	
Number of violations	The level of infeasibility of the roster w.r.t. the (hard) constraints
Nurse satisfaction	The percentage of nurse shift preferences that have remained intact
Utilisation	The percentage of the total assignable hours (w.r.t. nurse contracted hours h) already assigned over the whole roster and within the week of the violation
Local problem characteristics	
Nurse rank	An index assigned based on the qualification level of the nurse or NurseType involved in the violation
Nurse satisfaction	The percentage of shift preferences that have remained intact w.r.t. nurses of the type involved in the violation
Utilisation	The percentage of total assignable hours assigned of the nurses of the type involved in the violation over the whole roster and within the week of the violation

search whereby cases are either accepted or not and no similarities are evaluated. The equality of the $ViolationType$ variables described in Section 3 is used to make the case comparisons.

The second search ranks the restricted set of cases using the similarity function Sim applied to the violation indices. These indices are the characteristics of the problem identified as having an influence on the decision making process and are divided into two categories. Global characteristics are the properties of the roster as a whole including the number of violations, levels of staff satisfaction and current utilisation statistics. Local characteristics describe the problem in the location of the violation. They include the magnitude of the violation, and the current satisfaction and utilisation statistics of the types of nurses involved. Table 1 lists all the problem indices and gives a brief description of each.

Formally, the retrieval algorithm is described by the function Retrieve $(violation_\alpha, CB, CB')$ (see Figure 2) which from the case-base CB returns a set of cases, CB', sorted in order of similarity, that are compatible to the current problem $violation_\alpha$. Lines 1 and 2 initialise CB' and apply the function θ_v^R to $violation_\alpha$ to get the generalised form. The set of cases CB' is filled, in lines 3 to 7, with all cases in the case-base of the same $ViolationType$ as the current problem. By the definition of equality (as defined in Section 3) this includes the parameters of the violation. If there are no such cases then line 8 returns false

```
RETRIEVE ($violation_\alpha, CB, CB'$)
1: $CB' \leftarrow \emptyset$.
2: $\langle ViolationType_\alpha, ViolationIndices_\alpha \rangle \leftarrow \theta_v^R(violation_\alpha)$.
3: for all $case_\gamma = \langle\langle ViolationType_\gamma, ViolationIndices_\gamma \rangle, r_\gamma \rangle \in CB$ do
4:   if $ViolationType_\gamma = ViolationType_\alpha$ then
5:     $CB' \leftarrow CB' \cup \{case_\gamma\}$.
6:   end if
7: end for
8: if $|CB'| = 0$ then return $false$. end if
9: generate an array $score[|CB'|]$.
10: for all $case_\gamma = \langle\langle ViolationType_\gamma, ViolationIndices_\gamma \rangle, r_\gamma \rangle \in CB'$ do
11:   $score[\gamma] \leftarrow$ SIM$(ViolationIndices_\alpha, ViolationIndices_\gamma)$.
12: end for
13: sort $CB'$ according to $score$.
14: return $true$.
```

Fig. 2. Retrieval algorithm

and a manual solution will be required. Lines 9 to 13 generates an array of scores for each of the cases in CB' and then sorts CB' accordingly.

This sorted set of restricted cases generated by RETRIEVE is then passed to another method for repair adaptation. The adaptation process is also separated into two phases. Initially, the method generates, using the data from the current roster, a set of candidate repairs each of the same type as in the retrieved case. The second stage involves ranking these candidate repairs according to their similarity to the repair in the retrieved case. Here the set of repair indices from the retrieved case is compared with the calculated indices of the candidates using the SIM function. The exact indices that are used depend on the type of repair being generated. Table 2 lists the indices used for each of the three different repair types.

Formally, the function GENERATEREPAIR$(R, CB', violation_\alpha, Candidates)$ (see Figure 3) tries to generate a repair as similar as possible to that used in the retrieved cases. The array of possible repairs $Candidates$ is filled by a function DETERMINECANDIDATES. If there are no available candidates given the repair information from the retrieved case then the next case in the $Candidates$ array is considered. Lines 1 to 8 of the algorithm try to produce a set of candidates for each of the cases in CB' starting with the most similar. Then, analogously to the retrieval algorithm, the candidates are ranked according to their similarity to the repair in the retrieved case with respect to their $RepairIndices$.

The DETERMINECANDIDATES $(N, violation_\alpha, RepairType_0)$ function is the key to the generation of repairs. This returns a set of actual repairs that match the retrieved repair in terms of the type of the repair and the types of the nurses and shifts involved. Six sets of rules have been established for determining the candidates – one for each possible pair of violation and repair types. These rules are fairly intuitive and use some very simple domain knowledge to produce.

Table 2. Repair indices

Feature	CBREASSIGN	CBSWAP	CBSWITCH
# nurses assigned $newShift1$ on $day1_\beta$	√	√	√
# nurses assigned $newShift2$ on $day1_\beta$		√	√
# nurses assigned $newShift1$ on $day2_\beta$			√
# nurses assigned $newShift2$ on $day2_\beta$			√
# nurses of type $NurseType1_\beta$ assigned $newShift1$ on $day1_\beta$	√	√	√
# nurses of type $NurseType1_\beta$ assigned $newShift2$ on $day1_\beta$		√	√
# nurses of type $NurseType1_\beta$ assigned $newShift1$ on $day2_\beta$			√
# nurses of type $NurseType1_\beta$ assigned $newShift2$ on $day2_\beta$			√
# nurses of type $NurseType2_\beta$ assigned $newShift1$ on $day1_\beta$		√	
# nurses of type $NurseType2_\beta$ assigned $newShift2$ on $day1_\beta$		√	
# nurses assigned $oldShift$ on $day1_\beta$	√		
# nurses of type $NurseType1_\beta$ assigned $oldShift$ on $day1_\beta$	√		
Assigned/Contract Hours $nurse1_\beta$	√	√	√
Assigned/Contract Hours $nurse2_\beta$		√	
comp value for $nurse1_\beta$ around $day1$	√	√	√
comp value for $nurse1_\beta$ around $day2$			√
comp value for $nurse2_\beta$ around $day2$		√	

```
GENERATEREPAIR ($N, CB', violation_\alpha, Candidates$)
 1: $Candidates \leftarrow \emptyset$
 2: $index \leftarrow 0$
 3: while $|Candidates| = 0$ do
 4:    $case := \langle v, \langle RepairType, RepairIndices\rangle\rangle \leftarrow CB'[index]$.
 5:    $Candidates \leftarrow$ DETERMINECANDIDATES ($N, violation_\alpha, RepairType$).
 6:    $index \leftarrow index + 1$.
 7:    if $index = |CB'|$ then return false. end if
 8: end while
 9: generate an array $score[Candidates]$.
10: for all $repair_\beta \in Candidates$ do
11:    $\langle RepairType_\beta, RepairIndices_\beta\rangle \leftarrow \theta_r^R(repair_\beta)$.
12:    $score[repair_\beta] \leftarrow$ SIM($RepairIndices_0, RepairIndices_\beta$).
13: end for
14: sort $Candidates$ according to $score[Candidates]$.
15: return $true$.
```

Fig. 3. Adaptation algorithm

Nevertheless, they are quite large when notated and so here we shall only present the following rule, which shall be used in the example in Section 5:

Given $violation_\alpha = COVER(NurseType_\alpha, day_\alpha, shift_\alpha)$ and $repairType = CBREASSIGN(NurseType1, newShift1, oldShift)$, then $\textsc{DetermineCandidates}\left(N, violation_\alpha, repairType\right) = \left\{REASSIGN(nurse1_{i_\beta}, day1_\beta, shift_\beta) \middle| (nurse1_{i_\beta} \in N) \wedge (NurseType1_\beta = NurseType1) \wedge (s_{c,day_\alpha} = oldShift), day1_\beta = day_\alpha, shift_\beta = shift_\alpha\right\}$.

This rule returns a set of repairs all of which have the *REASSIGN* type with different parameter values. The nurse in each repair must be of the same type as that used in the repair from the retrieved case. In addition the shift currently assigned to each nurse on the day of the violation must be the same as the *oldShift* parameter of the retrieved repair. The *day* and *shift* parameters of the repair must be the same as that of the violation for this rule.

5 Example

In order to give an illustrative example of the method we shall consider a relatively simple problem solving episode. It is assumed that the weighting of all similarity calculations is flat (all weights equal 1). The problem we are attempting to solve is a cover violation of the roster R – that there is no registered nurse (RN) rostered on the early shift ($shift = E$) of the third day ($day = 2$) of the planning period. This is represented as

$$violation_\alpha = COVER(\{RN, 0, 0, 0, 0\}, 2, E) .$$

Here the zero-valued elements in the $NurseType$ set indicate that there is no restrictions on these feature values (so a suitable registered nurse could be male or female, specialty trained or untrained, etc.).

This violation is first passed to the Retrieve method. The generalised form of this violation is generated by the θ_v^R as follows:

$$\theta_v^R\left(violation_\alpha\right) = \langle CBCOVER(\{RN, 0, 0, 0, 0\}), ViolationIndices_\alpha\rangle,$$
where $ViolationIndices_\alpha = \{62.00, 99.46, 47.29, 54.32, 1.00, 78.41, 44.63, 50.76\}$.

Each of the values in the $ViolationIndices_\alpha$ array corresponds to one of the violation feature values described in Table 1. We must now find all cases $case_\gamma \in CB$ with $ViolationType = CBCOVER(\{RN, 0, 0, 0, 0\})$ and add them to the set of cases CB'. In this example we find three such cases. The Retrieve method then calculates the similarity between the $ViolationIndices_\gamma$ of each of these cases and the $ViolationIndices_\alpha$ array:

Case	Features	Sim (score)
α	62.00 99.46 47.29 54.32 1.00 78.41 44.63 50.76	NA
$CB'[0]$	57.00 92.31 48.12 52.14 1.00 80.32 74.23 90.32	8.749
$CB'[1]$	31.00 86.32 53.98 78.92 1.00 44.02 70.23 60.12	4.975
$CB'[2]$	80.00 83.24 70.34 80.23 2.00 70.12 80.12 85.23	3.997

Finally, the CB' set is sorted according to the similarity values in the far right column. This set now contains all compatible cases in order of the similarity between their and the current problem's violation features.

Now that a set of cases of similar problems has been identified, their corresponding solutions need to be adapted to the current problem $violation_\alpha$. For simplicity in this example we shall assume that the first case ($CB'[0]$) allows the production of such a set of candidates. The repair part of this case contains the following:

$$\begin{aligned} r = \langle & CBREASSIGN(\{RN, F, H, ET, E\}, E, U), RepairIndices\rangle, \\ & \text{where } RepairIndices = \langle \{11, 9, 2, 0, 59.1, 0\}, EEUUL\rangle\,. \end{aligned}$$

The GenerateRepair function fills the *Candidates* array with potential repairs using the DetermineCandidates function. By applying the conditions given in Section 4 for $CBREASSIGN$ repairs given a $COVER$ violation we get three candidate repairs, each of which uses a different registered nurse with $NurseType = \{RN, F, H, ET, E\}$. These candidate repairs are

$$\begin{gathered} REASSIGN(nurse_2, 2, E)\,; \\ REASSIGN(nurse_8, 2, E)\,; \\ REASSIGN(nurse_{15}, 2, E)\,. \end{gathered}$$

We apply the generalisation function θ_r^R to each of these candidate repairs and compare their resulting $RepairIndices$:

Repair	Features	Sim (score)
retrieved	11 9 2 0 59.1 $\textsc{Comp}_0(EEUUL) = 0$	NA
$\theta_r^R(Candidates[0])$	5 5 3 0 20.3 $\textsc{Comp}_\alpha(EELLO) = 3$	4.090
$\theta_r^R(Candidates[1])$	10 9 3 0 62.7 $\textsc{Comp}_\alpha(OEUUE) = 2$	9.357
$\theta_r^R(Candidates[2])$	10 9 3 0 84.0 $\textsc{Comp}_\alpha(UUUUU) = 3$	5.433

The candidate repair that is closest to the repair in the best case $CB'[0]$ is $Candidates[1]$ – which is a reassignment of $nurse_8$ to the EARLY shift on the third day of the planning period.

The example here is simpler than many encountered for ease of explanation. Other combinations of violation and repair types involve a more complex search for candidate repairs. However, all the principals needed for more complex instances are the same.

6 Results

The results presented in this section illustrate how the method performs at imitating the rostering decisions of humans. We do not compare performance with other rostering methods for two reasons. Firstly, the amount of information used and the way it is presented is incompatible with most existing problem formulations. More significantly, the case-based reasoning method here treats constraint violations as individual *problems* rather than providing solutions to entire rostering problems in the traditional sense.

The method has been implemented and tested on real-world data from the QMC. This data consisted of twelve 28-day rosters and the corresponding preference information for 19 nurses of various qualification and training levels. Nine constraints were defined consisting of eight cover type constraints detailing required skill mixes for the three shifts and one totals constraint limiting the number of hours in a fortnight to 75 per nurse.

Two sets of experiments were defined. The aim of the experiments was to determine the quality of the reasoning process in terms of the agreement between automated decisions and those of the nurse rostering expert. Constraint violations were identified at random and the repairs suggested by this method were compared to the repairs actually made in the final roster. These expert repairs were determined by comparing the final and preference rosters and only those instances where the decision was clearly evident were considered. The "quality" of a generated repair was assessed by comparing it with the expert repair and assigning one the following verdicts:

– Exact match: the generated repair is identical to the expert's repair;
– Equivalent match: the generated repair involves nurses of the same types and the same shifts as those used in the expert's repair;
– Fail: the generated repair is not an exact or equivalent match, or no repair was generated.

The first experiment involved repairing five runs of 120 constraint violations. Three repairs were suggested by the method and compared to the expert repair. The case-base is empty at the start of the run and the expert repair for each of the constraint violations is stored after it is applied to the roster. In this way the method is storing more experience in the case-base as the run progresses. Figure 4 shows the average cumulative number of exact and equivalent matches against the case-base size for each of the three suggested repairs. The bold lines are the first (or best with respect to the reasoning process) repairs for each iteration.

The results in Figure 4 show an increasing gradient of all lines indicating an increasing number of repairs of the given verdict per iteration. It can be seen from this that the case-base learns how to produce more exact or equivalent repairs as its size increases. An increase in the amount of training given to the case-base corresponds to an increase in the quality of the repairs produced. It is particularly encouraging that the first suggestions in general score more exact and equivalent matches than the second and third. The increases in solution quality are made more apparent in Figure 5. This shows the percentage of exact

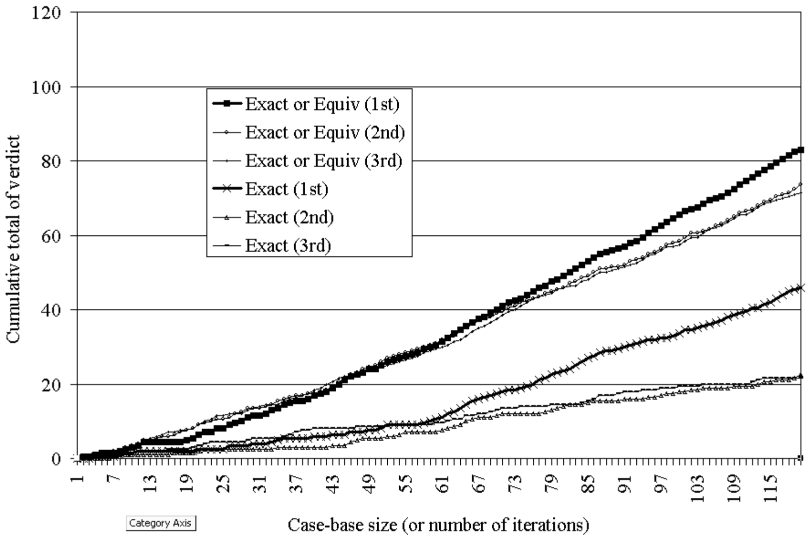

Fig. 4. Average cumulative number of exact and equivalent matches against case-base size over five 120 iteration runs

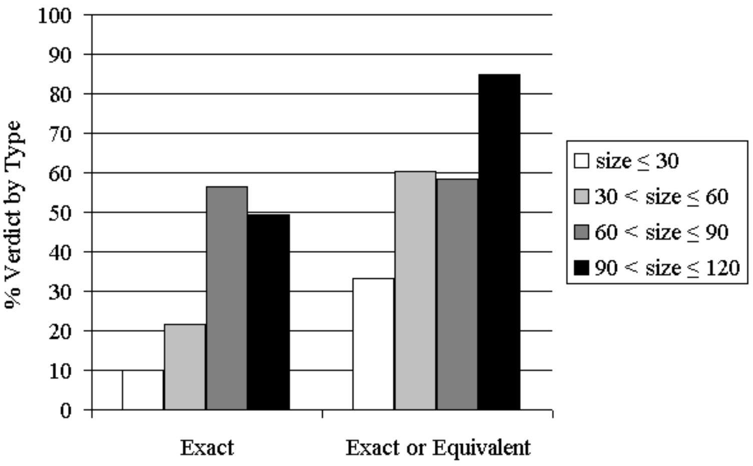

Fig. 5. Effects of case-base size on solution quality

and equivalent repairs at different stages in the runs. In general, in the later stages, when the case-base contains more experience, a larger number of good suggestions are produced.

The second set of experiments was defined to test the influence of different types of indices, namely global and local, on the reasoning process. Figure 6 shows the results of an experiment carried out using case-bases generated during

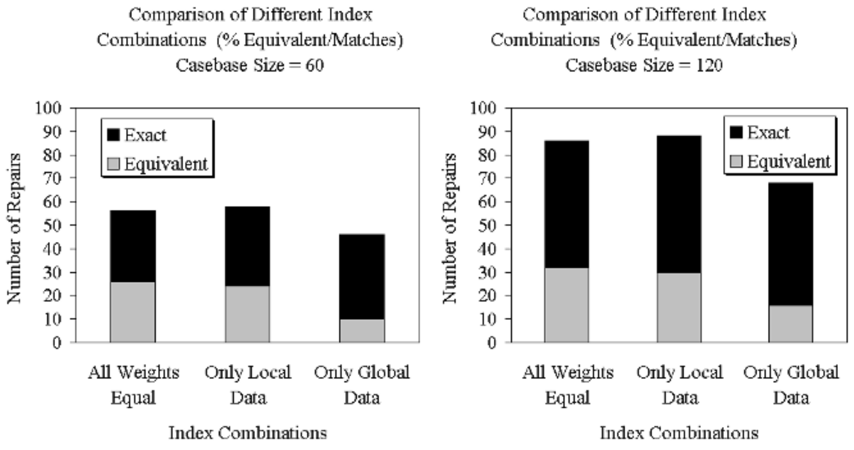

Fig. 6. Effects of different combinations of problem index on the solution quality

the previous test. Case-bases containing 60 and 120 cases were collected during each run. Using each of these 10 case-bases an additional 100 constraint violations were repaired but this time there was no storage of new cases. Three different combinations of problem index weights were set in the similarity function so that all indices, only local indices, and only global indices were used.

The lower-quality results from the only global data set suggest, at first, that the global indices are not useful. The best results were achieved when only local data was used. However, the global data set nevertheless produced a reasonable percentage of good results and this suggests that global information may still be useful for the reasoning process. It is certainly the case that global data are not as important as the local data and this should be reflected in any automated feature weighting that may be carried out in future work.

7 Conclusion

This paper has introduced a new approach to the employee rostering problem. The approach is different to existing methods in the sense that it does not use explicitly defined evaluation functions, which often fail to include all aspects of the rostering problem. The results show that it is possible to capture and imitate the rostering actions of human rostering experts. By storing the rules for repairing constraint violation implicitly in the case information we have created a technique that is both adaptable and flexible.

We intend to increase the number of different types of constraints that are considered including weekend constraints, night shift constraints, and possibly any soft constraints that are not adequately covered by the information in the case-base. This may involve adding additional structural and index elements to the definition of a case. The weighting of indices to reflect their relative importance in the reasoning process must be investigated. A system that dynamically allocates weights under different conditions is being considered. This allocation may depend on the content of the case-base and thus reflect the importance placed on indices by the expert.

The success of this violation/repair model for problem solving will be built on by incorporating the method within an intelligent iterative algorithm. A meta-heuristic could use the technique to decide its next move in the search space at each iteration. Although the processing time for each repair generation is negligible, the cumulative effect of multiple searches of the case-base and the impact on overall algorithm performance will need to be addressed. A hybrid algorithm could be evaluated and compared with existing rostering methods and its design and behaviour will be the subject of future research.

Acknowledgements. This research is supported by the Engineering and Physical Sciences Research Council (EPSRC) in the UK, grant no GR/N35205/01.

References

1. Abdennadher, S., Schlenker, H.: INTERDIP – an Interactive Constraint Based Nurse Scheduler. In: Proc. 1st Int. Conf. Exhib. Pract. Applic. Constraint Technol. Logic Program. (PACLP) (1999)
2. Arthur, J.L., Ravindran, A.: A Multiple Objective Nurse Scheduling Model. AIIE Trans. **13** (1981) 55–60
3. Bailey, J., Field, J.: Personnel Scheduling with Flexshift Models. J. Oper. Manage. **5** (1985) 327–338
4. Beaumont, N.: Scheduling Staff Using Mixed Integer Programming. Eur. J. Oper. Res. **98** (1997) 473–484
5. Berrada, I., Ferland, J.A., Michelon, P.: A Multi-objective Approach to Nurse Scheduling with Both Hard and Soft Constraints. Socio-Econ. Planning Sci. **30** (1996) 183–193
6. Burke, E.K., De Causmaecker, P., Vanden Berghe, G.: A hybrid tabu search algorithm for the nurse rostering problem. In: Proc. 2nd Asia Pacific Conf. on Simulated Evolution and Learning (selected papers). Lecture Notes in Artificial Intelligence, Vol. 1585. Springer-Verlag, Berlin Heidelberg New York (1998) 187–194
7. Burke, E.K., Cowling, P.I., De Causmaecker, P., Vanden Berghe, G.: A Memetic Approach to the Nurse Rostering Problem. Applied Intelligence. Kluwer, Dordrecht (2001) 199–214
8. Cheng, B.M.W., Lee, J.H.M., Wu, J.C.K.: A Constraint-Based Nurse Rostering System Using a Redundant Modeling Approach. Department of Computer Science and Engineering, The Chinese University of Hong Kong (1996)
9. Dowsland, K.A., Thompson, J.M.: Solving a Nurse Scheduling Problem with Knapsacks, Networks and Tabu Search. J. Oper. Res. Soc. **51** (2000) 825–833
10. Dowsland, K.: Nurse Scheduling with Tabu Search and Strategic Oscillation. Eur. J. Oper. Res. **106** (1998) 393–407
11. Kolodner, J.L.: Case-Based Reasoning. Morgan Kaufmann, San Mateo, CA (1993)

12. Meisels, A., Gudes, E., Solotorevski, G.: Employee Timetabling, Constraint Networks and Knowledge-Based Rules: A Mixed Approach. In: Burke, E, Ross, P. (Eds.): Practice and Theory of Automated Timetabling I (PATAT 1995, Edinburgh, Aug/Sept, selected papers). Lecture Notes in Computer Science, Vol. 1153. Springer-Verlag, Berlin Heidelberg New York (1996) 93–105
13. Meyer auf'm Hofe, H.: Solving Rostering Tasks as Constraint Optimization. In: Burke, E, Erben W. (Eds.): Practice and Theory of Automated Timetabling III (PATAT 2000, Konstanz, Germany, August, selected papers). Lecture Notes in Computer Science, Vol. 2079. Springer-Verlag, Berlin Heidelberg New York (2001) 280–297
14. Miller, M.L., Pierskalla, W., Rath, G.: Nurse Scheduling Using Mathematical Programming. Oper. Res. **24** (1976) 857–870
15. Miyashita, K., Sycaras, K., Mizoguchi, R.: Capturing Scheduling Knowledge from Repair Experiences. Int. J. Human-Comput. Stud. **41** (1994) 751–773
16. Scott, S., Simpson, R.: Case-bases Incorporating Scheduling Constraint Dimensions – Experiences in Nurse Rostering. In: Smyth, B., Cunningham, P. (Eds.): Advances in Case-Based Reasoning (EWCBR'98). Lecture Notes in Computer Science, Vol. 1488. Springer-Verlag, Berlin Heidelberg New York (1998) 392–401
17. Silvestro, R., Silvestro, C.: An Evaluation of Nurse Rostering Practices in the National Health Service. J. Adv. Nursing **32** (2000) 525–535
18. Warner, M.: Scheduling Nursing Personnel According to Nurse Preference: a Mathematical Programming Approach. Oper. Res. **24** (1976) 842–856