A novel approach to finding feasible solutions to personnel rostering problems

Gareth Beddoe and Sanja Petrovic

Automated Scheduling, Optimisation, and Planning Research Group,
School of Computer Science and Information Technology, University of Nottingham,
Jubilee Campus, Nottingham NG8 1BB, United Kingdom
{ grb | sxp }@cs.nott.ac.uk

Abstract. Classical meta-heuristic methods for solving rostering problems focus on defining measures of roster quality. Here we present a new case-based reasoning approach to generating repairs of hard constraint violations using expert-human experience. This approach is used to guide heuristic constraint satisfaction algorithms, eliminating the need to explicitly define search objectives.

1 Introduction

Employee rostering problems are often highly constrained and difficult to solve manually [9]. Legal, management, operational, and staff requirements are often conflicting and must be taken into account when making rostering decisions. For example, management requirements for cover and skill mix conflict with the maximum working hours allowed (by law and contract) as well as individual staff preferences. Nevertheless, rostering experts develop strategies for balancing these conflicts and are able to solve most problems. This paper describes a constraint satisfaction approach to automated rostering that draws on stored experience of human rostering experts.

A number of different operational research approaches have been explored for solving employee rostering problems using techniques including linear and integer programming [3, 5, 20, 23], goal programming [2, 6], and constraint satisfaction techniques [1, 9, 14, 15]. Scott and Simpson [22] combined case-based reasoning and constraint logic programming by storing shift patterns used for the construction of rosters. CBR has been successfully used for machine scheduling problems in [17, 21]. A number of meta-heuristic methods have been developed with some success using methods such as (but not limited to) tabu search [7, 10, 11], simulated annealing and genetic algorithms [4], and memetic algorithms [8]. These methods explore the search space through neighbourhoods of solutions 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 the specific constraint violations found in individual problems.

Case-based reasoning (CBR) [13] is an artificial intelligence methodology that imitates 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 the most similar to the current problem in terms of a set of characteristic *indices*. The solutions from these retreived cases are then adapted to the context of the current problem. If a new solution might be useful for solving problems in the future then it is stored as a new case in the case-base.

CBR is used here for the rostering problem as a means by which to capture the rostering knowledge of human experts by storing a history of constraint violations and their corresponding repairs [19]. The rostering problem is presented as a constraint satisfaction problem whereby a roster is defined as feasible if it does not violate any of a set of hard constraints. Two iterative heuristic constraint satisfaction algorithms are presented that attempt to find feasibility by direct repair of individual constraint violations within the roster. The first algorithm repairs constraint violations randomly within a tabu search framework while the second uses the rostering experience stored within the case-base. A third, hybrid, approach uses tabu-lists and case-based repair generation. A number of experiments were designed to investigate the performance of these algorithms.

The problem investigated here is that of rostering nurses in the Ophthamological Ward at the Queens Medical Centre University Hospital Trust (QMC) in Nottingham, United Kingdom. In Section 2 the rostering problem at the QMC is formulated and in Section 3 an overview of the case-based reasoning framework described in [18, 19] is given. The constraint satisfaction algorithms are described in Section 4 incorporating case-based repair generation and tabu-list concepts. Sections 5 and gives the results of experiments using the real-world data from the QMC. A discussion of the benefits of the methods proposed is given in Section 6.

2 Problem Formulation

We define the nurse rostering problem as the ordered pair:

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

where $N = \{nurse_i : 0 \leq i < n\}$ is the set of n nurses to be rostered and C is a set of constraints.

Nurses in the QMC ward have one of four qualification levels. 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 practical and management training whereas ENs are mainly trained in the practical aspects of nursing. ANs are unqualified nurses who can perform basic duties and SNs are currently training to be either RNs or ENs. These real nurse types are grouped here using three additional qualification types. The classification XN denotes a nurse of any type. RNs, ENs, and ANs are grouped together as employed (PN) and RNs and ENs as qualified (QN). In addition to these basic qualifications nurses can also recieve ward-specific training (in this case eye training - denoted ET) and a grade, ranging from A to I, is determined by a combination of qualification, specialty training, and the amount of practical experience they have. The gender ((M)ale or (F)emale) and international status ((I)nternational or (H)ome) of nurses is also taken into account when making rostering decisions.

Here a nurse is denoted by the 4-tuple:

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

where $NurseType_i = \{f_{i,1} \dots f_{i,5}\}$ is an array of descriptive features where

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

is the nurse's qualification. The QN, PN, and XN qualifications are included because the *NurseType* variable is also used to describe general skill requirements within constraints and violations. The remaining elements, $f_{i,2} \dots f_{i,5}$, describe the gender, international status, specialty training, and grade of the nurse. The number of hours the nurse is contracted to work in a week (normally 37.5) is denoted by $hours_i \in \mathbb{R}^+$.

The set of assignment variables:

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

represents the shift assignments for $nurse_i$ on day j over the number of days for which the roster has been constructed, denoted by *period*.

The set of variables:

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

represents the preferred shift assignments of $nurse_i$ on day j. Preferred assignments are established by a process of pre-rostering consultation and are used as the initial instantiation of the shift assignment variables (i.e. initially $NR_i = PR_i \forall i$). 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 (hard) constraints of the following types:

Cover constraints define the skill mixes required for different shifts. A minimum number of nurses of a particular feature set (represented by the $NurseType$ variable) must be assigned to particular shifts. For example, the early shift requires 4 QN's;
MaxDaysOn constraints limit the number of days that nurses may work in a row. For the QMC ward this is generally 5 for all nurses;
MinDaysOn constraints define the minimum number of days that nurses may work succesively;
Succession constraints define illegal shift combinations for nurses matching a feature set. A $NIGHT$ shift followed by an $EARLY$ shift is one such combination;
Totals constraints define the maximum number of hours a nurse may work over a set time period. For example, no nurse may work more than 75 hours in a fortnight.

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 constraint violations in R and the possible repairs to the current roster R. An element $violation_\alpha \in P_v^R$ stores the type of violation and the parameters relevant to it. Violations are denoted as follows:

CoverViolation($NurseType, day, shift$)**:** There are insufficient nurses of feature set $NurseType$ assigned to $shift$ on day;
MaxDaysOnViolation($nurse_i, startDay, numDays$)**:** $nurse_i$ is working too many shifts in a row starting on $startDay$ for $numDays$ days;
MinDaysOnViolation($nurse_i, startDay, numDays$)**:** $nurse_i$ is working too few shifts in a row starting on $startDay$ for $numDays$ days;
SuccessionViolation($nurse_i, day$)**:** $nurse_i$ has an illegal shift combination on days day and $day + 1$;
TotalsViolation($nurse_i, startDay, endDay$)**:** $nurse_i$ is working 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($nurse_i, day, shift$)**:** Assign $shift$ to $nurse_i$ on day;
Swap($nurse_i, nurse_j, day$)**:** Interchange the shift assignments of $nurse_i$ and $nurse_j$ on day;
Switch($nurse_i, day1, day2$)**:** Interchange the shift assignements of $nurse_i$ between days $day1$ and $day2$.

The sets P_v^R and P_r^R 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 - one that is independent of any individual rostering problem instance.

We define the case-base CB as a set of previously encountered violations and their corresponding repairs. Figure 1 gives a graphical representation of the variables described in the remainder of this section, and of the repair generation technique described in Section 3. This figure emphasises the difference between variables that pertain to the current rostering problem and those generalised variables that must interact with the historical information in the case-base.

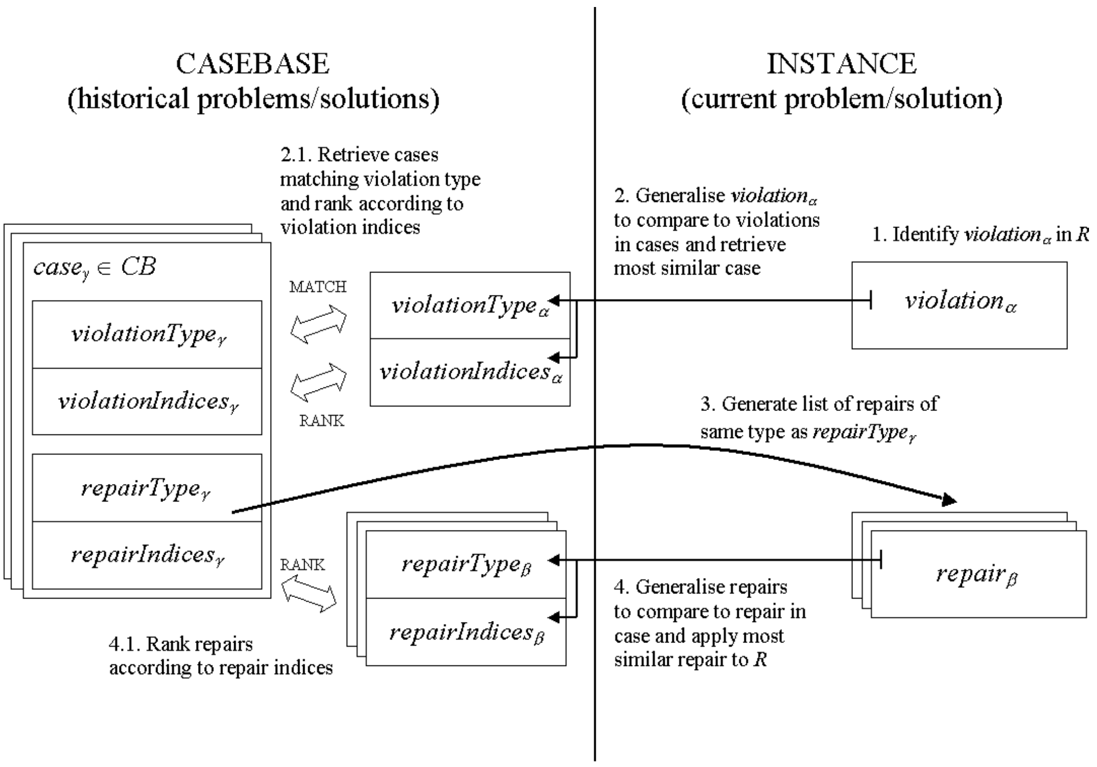
CASEBASE
(historical problems/solutions)
INSTANCE
(current problem/solution)
2.1. Retrieve cases matching violation type and rank according to violation indices
2. Generalise violation_α to compare to violations in cases and retrieve most similar case
1. Identify violation_α in R
case_γ ∈ CB
violationType_γ
violationIndices_γ
repairType_γ
repairIndices_γ
MATCH
RANK
violationType_α
violationIndices_α
violation_α
3. Generate list of repairs of same type as repairType_γ
RANK
repairType_β
repairIndices_β
repair_β
4. Generalise repairs to compare to repair in case and apply most similar repair to R
4.1. Rank repairs according to repair indices

Fig. 1. A graphical summary of the repair generation technique. After application of the repair a new case can be stored using the generalised verions of both the violation and repair.

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 $\gamma \in \Gamma$. 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 violations and repairs independently of any problem instance. $ViolationIndices_\gamma$ and $RepairIndices_\gamma$ store the feature information needed to identify similar problem instances during case retrieval, and to generate repairs during case adaptation. The index sets are arrays of statistical feature values. They are the characteristics of the problem identified as having an influence on the decision making process. The violation indices include measures of roster infeasibility, nurse utilisation and satisfaction, shift pattern statistics, and shift coverage, depending on the specific type of violation. The repair indices are statistical information about the parameters of the repair - the cover of shifts before and after the repair and the utilisation and satisfaction of the nurses involved.

3 Case-based Repair Generation

The method described in this section generates repairs of constraint violations within a roster using the generalised experience stored in the case-base. Key to the success of this method is the notion of *problem similarity.* The method must first find the most similar problem in the case-base to the current problem (i.e. the violation within the current roster). Having identified the most similar violations the corresponding repairs must be adapted to the context of the current roster. Using the notion of *solution similarity* and a set of simple rules, repairs are generated that are as similar to the retrieved solution as possible. Following are the descriptions of the retrieval process, whereby similar violations are identified, and the adaptation process for generating repairs.

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. The second search ranks the restricted set of cases using the similarity function SIM defined as follows:

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

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

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

where $index_{max_i}$ and $index_{min_i}$ are the maximum and minimum values of index i over the whole case-base.

Hence, given a problem $violation_\alpha$, the retrieval algorithm comprises of these steps:

1. Generalise $violation_\alpha$ to get $v_\alpha = \langle ViolationType_\alpha, ViolationIndices_\alpha \rangle$ by generalising the parameter information and calculating the violation indices with respect to the current roster;
2. Choose cases $case_\gamma = (v_\gamma, r_\gamma) \in CB$ such that:
$$ViolationType_\alpha = ViolationType_\gamma \ ;$$
3. Rank these cases according to $\textsc{Sim}(ViolationIndices_\alpha, ViolationIndices_\gamma)$.

This sorted set of cases is then passed to the adaptation process which is also seperated into two phases. Initially, the method generates, using the data from the current roster, a set of candidate repairs 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 indices of the candidate repairs using the SIM function.

The adaptation algorithm can be summarised as follows:

1. Consider the top ranking case $case_0 = (v_0, r_0)$;
2. Generate a set of repairs of type $RepairType_0$ using all possible relevant combinations of nurses and shifts from the roster. If no repairs can be generated then remove the top case and goto 1;

3. Generalise this set by calculating the $RepairIndices_\beta$ for each $repair_\beta$;
4. Rank the repairs according to $\textsc{Sim}(RepairIndices_\beta, RepairIndices_0)$.

To illustrate both these processes we shall describe a simple example (see the Appendix for additional details). Consider the following roster (where E = EARLY, L = LATE, and U = UNASSIGNED):

	0	1	2	3	4	5	6
$nurse_0$	E	U	U	U	L	E	E
$nurse_1$	L	L	L	U	U	L	U
$nurse_2$	U	E	E	L	E	U	L

Here $nurse_0$ and $nurse_1$ are eye-trained, registered, female, non-international, E-grade nurses and $nurse_2$ is an eye-trained, enrolled, female, non-international, D-grade nurse. Hence, $NurseType_0 = NurseType_1 = \{RN, F, H, ET, E\}$ and $NurseType_2 = \{EN, F, H, ET, D\}$.

A single constraint is applied to the roster, requiring that a minimum of 1 qualified nurse be assigned to every early shift. It can be seen that on day 3 there is no nurse assigned to the early shift. Therefore the violation $violation_\alpha = COVER(\{QN, 0, 0, 0, 0\}, 1, EARLY)$ needs to be repaired (where 0 in the feature set indicates that a feature can take any value).

The violation is passed to the retrieval process where all the examples of cover violations involving qualified nurses are identified in the case-base. Suppose, for this example, that three such cases are found. These must then be ranked according to the similarity measure as follows:

	Index Values								$\textsc{Sim}$ (score)
	$\lvert P_v^R \rvert$	GSat	GUO	GUW	Mag	LSat	LUO	LUW	
α	1	100.0	86.7	86.7	1	100.0	86.7	86.7	NA
$case_0$	1	92.3	89.2	78.3	1	93.0	82.3	78.1	2.53
$case_1$	2	86.2	73.9	68.9	1	44.0	80.2	80.1	1.36
$case_2$	15	60.8	58.3	45.6	2	70.1	80.1	60.2	0.77

The repair information stored in $case_0$ must now be adapted to create a repair of the current violation. For this example $case_0$ contains a (generalised) repair of type Reassign that uses a nurse with $NurseType = \{RN, F, H, ET, E\}$ who was originally assigned an UNASSIGNED shift on the day of the repair. The adaptation process generates a set of reassign repairs using nurses with the correct feature set and who are currently assigned UNASSIGNED on day 3. From our roster we have two nurses with such a set and so two repairs are generated:

$$repair_a = REASSIGN(nurse_0, 3, EARLY) \ ;$$
$$repair_b = REASSIGN(nurse_1, 3, EARLY) \ .$$

We generalise these repairs by calculating index sets and compare them to the repair from $case_0$:

repair	Index Values						$\textsc{Sim}$ (score)
	SCOA	SCOT	SCNA	SCNT	Util	SP	
$RepairIndices_0$	0	0	3	2	85.0	0	NA
$repair_a$	0	0	2	2	80.0	1	6.00
$repair_b$	0	0	2	2	100.0	4	2.67

Therefore the first repair, $REASSIGN(nurse_0, 3, EARLY)$, is returned as a solution. This current problem solving episode can be added to the case-base by generalising both the violation and repair, thus increasing the amount of experience stored.

4 Case-based Repair Generation for Constraint Satisfaction

In this section we shall describe three different heuristic constraint satisfaction algorithms. Two of these algorithms will make use of the case-based repair generation methodology presented in the previous section. Initially we have a roster that consists solely of the nurse's individual shift preferences. This roster violates a large number of constraints and the goal of the algorithms presented here is to find a feasible solution - i.e. one which does not violate the stated constraints.

The three constraint satisfaction algorithms attempt to find feasible solutions by succesively picking random violations and generating repairs. This approach differs from other heuristic constraint satisfaction techniques which generally search for feasibility by considering when particular instantiations of variables violate one or more constraints (e.g. [16]). The algorithms described here are not intended to compete with other constraint satisfaction methods. We shall simply use them to show the benefits that case-based repair generation can bring to a very simple algorithm.

The process of generating repairs does not include any explicit measure of roster quality insofar as it does not attempt to minimise the number of violations in the roster after application (or indeed to minimise the damage caused by repairs that generate new constraint violations). These algorithms are therefore performing a search for feasibility without any explicit representation of a search objective in the form of a function or otherwise.

We shall compare three different constraint satisfaction algorithms here. These algorithms differ in the way they generate and choose repairs, and in the search diversification methods that they employ. The first addresses constraint violations by randomly generating repairs and utilises a tabu-list to avoid local optima. The second uses the case-based retrievel and adaptation processes to repair randomly chosen violations and the third combines the case-based and tabu list approaches.

4.1 Random Repair Generation with Tabu List (RRGTL)

This algorithm uses no problem solving knowledge when generating repairs but uses the idea of tabu search proposed by Glover in [12]. A tabu-list of repairs is used to help the algorithm avoid local optima in the number of constraint violations and a tenure is specified which sets the length of the list (and therefore the number of iterations for which a stored repair will be considered 'tabu'). Some help is given to the algorithm by ensuring that the parameters of the violations, including the nurses, days, and shifts involved, are also included as parameters of the repairs. Otherwise the choice of repair type (Reassign, Swap, or Switch) and the other parameters involved is entirely random. No evaluation of the quality of repairs or the degree of violation of the roster is used when deciding on repairs.

Given roster $R = \langle N, C \rangle$ and tabu list T with tenure t :

```
1. Generate $P_v^R$ by applying the constraints in $C$ to $N$ ;
2. Pick random element $violation_\alpha \in P_v^R$ ;
3. Randomly create $repair_\beta$ using parameters of $violation_\alpha$ as appropriate ;
4. If $repair_\beta \notin T$ Then apply $repair_\beta$ to $N$ Else goto 3 ;
5. Add $repair_\beta$ to $T$ and enforce tenure $t$ ;
6. Generate $P_v^R$ by applying the constraints in $C$ to $N$ ;
7. If $|P_v^R| = 0$ Then exit Else goto 2 .
```

4.2 Case-Based Repair Generation (CBRG)

Here the experience stored in a case-base is used to generate repairs. It is assumed, for the experiments that follow, that the case-base has been well trained and contains sufficient examples of a variety of different problem solving episodes. Again there is no objective function used to choose repairs. The most similar repair from the most similar case is used at every iteration. There is no method for diversification of the search in this algorithm.

Given roster $R = \langle N, C \rangle$:

1. Generate P_v^R by applying the constraints in C to N ;
2. Pick random element $violation_\alpha \in P_v^R$;
3. Generate $repair_\beta$ using the case-based retrieval and adaptation methods ;
4. Apply $repair_\beta$ to N ;
5. Generate P_v^R by applying the constraints in C to N ;
6. If $|P_v^R| = 0$ Then exit Else goto 2 .

4.3 Case-Based Repair Generation with Tabu List (CBRGTL)

This hybrid algorithm combines aspects of the previous two algorithms. The RRGTL algorithm is not guided by any rostering knowledge as the repairs generated for each violation are random. CBRG guides the search using the knowledge in the case-base but is unable to cope when violations are repeatedly created - it will create the same repair for the violation each time it is encountered. The diversification provided by the tabu-lists and the rostering knowledge stored in the case-base are combined for the CBRGTL algorithm.

The algorithm includes an additional method of diversification. As well as storing a tabu list of repairs it is possible to store cases to prevent their re-use within a specified tenure. When a violation is encountered the method will retrieve the most similar case from the case-base and then generate a repair. If the same repair has been carried out before then this is an indication that the violation has reappeared during the repair of another violation. The tabu lists of repairs and cases help the algorithm to find feasibility by avoiding situations where the search loops between a number of conflicting constraints.

Given roster $R = \langle N, C \rangle$ and tabu lists $TRepair$ and $TCase$ with tenures tr and tc respectively:

1. Generate P_v^R by applying the constraints in C to N ;
2. Pick random element $violation_\alpha \in P_v^R$;
3. Retrieve the most similar case $case_0$;
4. If $case_0 \in TCase$ Then discard $case_0$ and goto 3 ;
5. Generate $repair_0$ from $case_0$ such that $repair_0 \notin TRepair$;
6. Add $case_0$ to $TCase$ and $repair_0$ to $TRepair$ and enforce tenures tr and tc;
7. Apply $repair_0$ to N ;
8. Generate P_v^R by applying the constraints in C to N ;
9. If $|P_v^R| = 0$ exit Else goto 2 .

5 Results

The three algorithms were tested on real-world data from the QMC using fourteen different variants:

- Case-based repair generation with no tabu lists (CBRG);
- Case-based repair generation with tabu lists of repairs with tenures 10, 20, and 50 (CBRGTL-R10, CBRGTL-R20, CBRGTL-R50);
- Case-based repair generation with tabu lists of cases with tenures 2, 5, and 10 (CBRGTL-C2, CBRGTL-C5, CBRGTL-C10);

Table 1. Algorithm Performance

	$\lvert P_v^R \rvert$ *best*	$\lvert Iterations \rvert$	$NSat$ (%)	$\#Feasible(/10)$
CBRG	1.9	382	93.36	0
RRGTL-R10	2.7	462	63.54	7
RRGTL-R20	5.3	499	60.43	4
RRGTL-R50	3.4	498	61.96	4
CBRGTL-R10	0	196	93.08	10
CBRGTL-R20	0	106	95.43	10
CBRGTL-R50	0	96	95.94	10
CBRGTL-C2	0.1	94	92.02	9
CBRGTL-C5	0	89	91.55	10
CBRGTL-C10	0	105	92.46	10
CBRGTL-C2-R10	0	88	92.56	10
CBRGTL-C5-R10	0	92	90.99	10
CBRGTL-C2-R20	0	92	92.74	10
CBRGTL-C5-R20	0	86	91.71	10

– Case-based repair generation with tabu lists of cases with tenures 2 and 5, and tabu lists of repairs with tenures 10 and 20, in all combinations (CBRGTL-C2-R10, CBRGTL-C5-R10, CBRGTL-C2-R20, CBRGTL-C5-R20);
– Random repair generation with tabu lists of repairs with tenures 10, 20, and 50 (RRGTL-R10, RRGTL-R20, RRGTL-R50).

A case-base was trained using 200 examples of violations and repairs derived from preference and final rosters over two four-week periods. A seperate four week preference roster was then used as the new problem to be solved for each of the runs. This initial roster contained 62 violations of 10 different constraints. The number of iterations was limited to 500 and the best solutions (i.e. those with the least constraint violations) were recorded. Table 1 shows the average results of 10 runs of each algorithm variant.

The first column of results in Table 1 shows the average number of constraint violations in the best solution that was found in each the 10 runs of each algorithm. A value of 0 indicates that feasibility was found on every run. The number of iterations needed to get the best solution and the percentage of nurse shift preferences satisfied are shown in the second and third columns. The number of feasible solutions found by each algorithm is shown in the final column

These results show very clearly the benefits of the hybrid algorithm. The RRGTL algorithm did not reliably find feasibility within 500 iterations for any tabu tenure although if allowed to continue it was successful eventually. The case-based repair generation algorithm with no tabu lists found a reasonable result comparatively quickly but was unable to do better irrespective of any additional time it was given. Invariably, this algorithm got 'stuck' in a repeating loop of violations that could not be resolved due to the reasons given in the previous section. All of the hybrid CBRGTL variants found feasibility with excellent reliability indicating that the combination of repair strategy from the case-base and diversification from the tabu-lists overcame the difficulties that the other two algorithms experienced on their own. Figure 2 shows the average performance of the three best variants in each class (CBRG, CBRGTL-C5-R20, and RRGTL-R10) against the number of iterations.

The real value of the hybrid approach can be seen by the nurse satisfaction performance. The expert trained case-base stored examples of repairs of constraint violations that attempted to reduce the violation of nurses individual shift preferences. The random repair generation method did not take this into consideration and so could not be expected to perform well on this criterion. More than one third of the nurses preferences were not satisfied in each of the best solutions

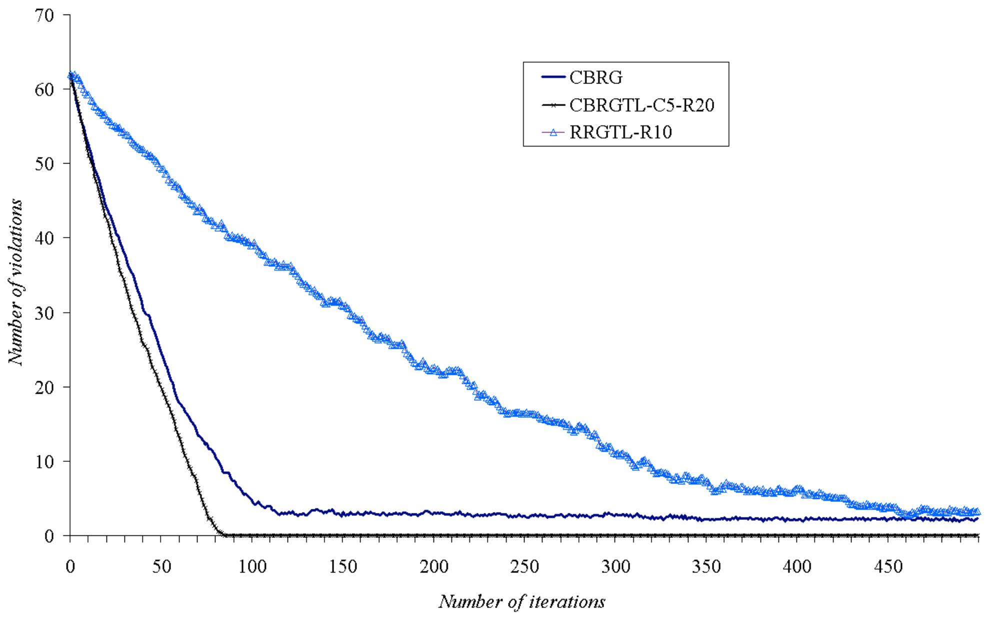

Fig. 2. Degree of violation vs. number of iterations (Max 500 Iterations)

for the tabu search variants (RRGTL). All of the case-based variants (CBRG and CBRGTL) performed excellently with respect to nurse satisfaction because of the implicit rostering knowledge stored in the case-base. Figure 3 shows the average nurse satisfaction performance over 500 iterations.

The results presented here are not intended to show that the case-based approach can outperform methods that employ a tabu search strategy. The RRGTL method shown here is by no means optimised for the problem - any sensible implementation would at least include explicit rules for avoiding violation of nurse preference under specific conditions and would probably chose repairs by minimising the number of new violations created. However, the results do show that when combined with a very simple algorithm, case-based repair generation can increase performance significantly.

6 Conclusion

In this paper a method for combining case-based reasoning with a meta-heuristic algorithm has been introduced. A case-base is used to capture expert rostering knowledge by storing examples of constraint violations and their corresponding repairs. A simple heuristic constraint satisfaction algorithm using tabu lists is enhanced by the knowledge stored in the case-base, leading to a significant performance improvement.

In fact the benefits of a case-based approach are threefold. The expert-quality repair examples stored in the case-base help the tabu search find feasibility much faster because they guide the search in sensible directions. The repairs in the case-base avoid violating nurse shift preferences wherever possible and so guide the search towards feasible solutions with high nurse satisfaction. Finally, all of this information is stored implicitly by the case-base and therefore does not need to be hard-coded into an algorithm using explicit rostering rules or objective functions.

Future research will focus on improving the knowledge capturing ability of the case-base. Methods to maintain the consistency of the knowledge and to identify erroneous or bad quality

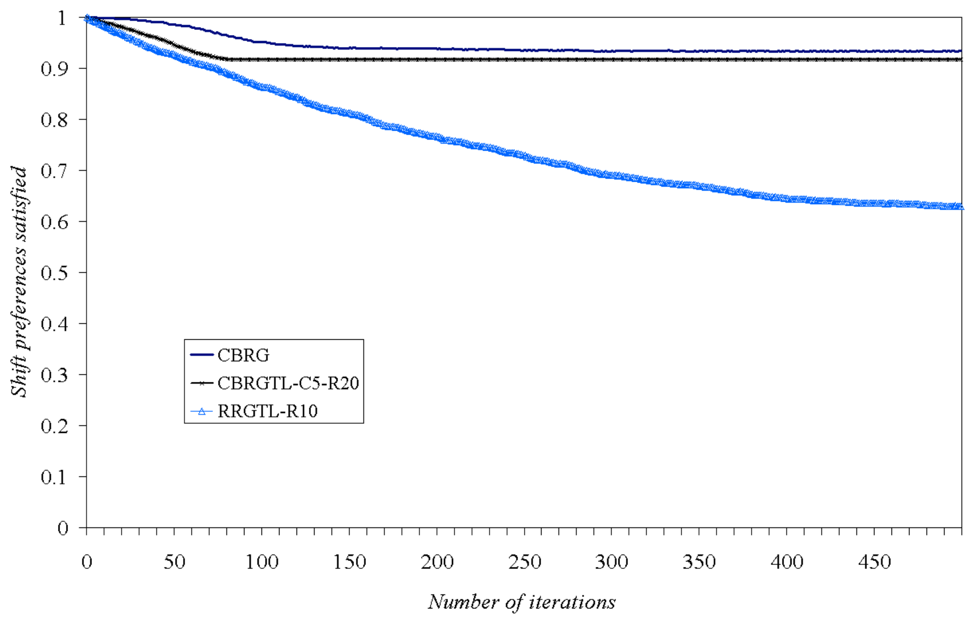

Fig. 3. Shift preference satisfaction vs. number of iterations (Max 500 Iterations)

data must be developed. The incorporation of the case-based repair generation methodology into other constraint satisfaction techniques will also be considered.

7 Acknowledgements

This research is supported by the Engineering and Physical Sciences Research Council (EPSRC) in the UK, grant number GR/N35205/01.

8 Appendix

The retrieval indices for the example in Section 3 are:

- $|P_v^R|$ - The number of constraint violations in R;
- GSat - The percentage of shift preferences satisfied for all nurses;
- GUO - The percentage of available working hours assigned for all nurses over the whole roster;
- GUW - The percentage of available working hours assigned for all nurses over the week containing the violation;
- Mag - The magnitude of the violation (in this case this is the cover shortage);
- LSat - The percentage of shift preferences satisfied for nurses with the $NurseType$ described in the violation;
- LUO - The percentage of available working hours assigned for nurses with the $NurseType$ described in the violation over the whole roster;
- LUW - The percentage of available working hours assigned for nurses with the $NurseType$ described in the violation over the week containing the violation.

The adaptation indices for example are:

– SCOA - The number of nurses of all types assigned to the original shift on the day of the repair;
– SCOT - The number of nurses of the $NurseType$ described in the repair assigned to the original shift on the day of the repair;
– SCNA - The number of nurses of all types assigned to the new shift on the day of the repair;
– SCNT - The number of nurses of the $NurseType$ described in the repair assigned to the new shift on the day of the repair;
– Util - The number of available hours assigned to the nurse in the repair;
– SP - A shift pattern distance score (described in detail in [19]).

The *original* shift is the shift assigned to the nurse before the repair was applied whilst the *new* shift was the shift assigned by the repair.

NB. Due to the small size of the example problem some of the indices described here may seem trivial. In larger, more complex problems this is not the case.

References

1. S Abdennadher and H Schlenker. INTERDIP – an interactive constraint based nurse scheduler. In *Proceedings of The First International Conference and Exhibition on The Practical Application of Constraint Technologies and Logic Programming, PACLP*, 1999.
2. J L Arthur and A Ravindran. A multiple objective nurse scheduling model. *AIIE Transactions*, 13(1):55–60, 1981.
3. J Bailey and J Field. Personnel scheduling with flexshift models. *Journal of Operations Management*, 5(3):327–338, 1985.
4. R N Bailey, K M Garner, and M F Hobbs. Using simulated annealing and genetic algorithms to solve staff scheduling problems. *Asia-Pacific Journal of Operational Research*, 14:27–43, 1997.
5. N Beaumont. Scheduling staff using mixed integer programming. *European Journal of Operational Research*, 98:473–484, 1997.
6. I Berrada, J A Ferland, and P Michelon. A multi-objective approach to nurse scheduling with both hard and soft constraints. *Socio-Economic Planning Sciences*, 30/3:183–193, 1996.
7. E K Burke, P De Causmaecker, and G Vanden Berghe. A hybrid tabu search algorithm for the nurse rostering problem. In *Selected Papers from the 2nd Asia Pacific Conference on Simulated Evolution and Learning Volume*, volume 1585 of *LNAI*, pages 187–194. Springer Verlag, 1998.
8. E K Burke, P I Cowling, P De Causmaecker, and G Vanden Berghe. A memetic approach to the nurse rostering problem. *Applied Intelligence*, pages 199–214, 2001.
9. B M W Cheng, J H M Lee, and J C K Wu. A constriant-based nurse rostering system using a redundant modeling approach. Technical report, Department of Computer Science and Engineering at The Chinese University of Hong Kong, 1996.
10. K A Dowsland and J M Thompson. Solving a nurse scheduling problem with knapsacks, networks and tabu search. *Journal of the Operational Research Society*, 51:825–833, 2000.
11. Kathryn Dowsland. Nurse scheduling with tabu search and strategic oscillation. *European Journal of Operational Research*, 106:393–407, 1998.
12. F Glover. Tabu search - part i. *ORSA Journal of Computing*, 1:190–206, 1989.
13. J L Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publishers Inc., 1993.
14. A Meisels, E Gudes, and G Solotorevski. Employee timetabling, constraint networks and knowledge-based rules: A mixed approach. In *Practice and Theory of Automated Timetabling, First International Conference*, pages 93–105. Springer, 1995.
15. H Meyer auf'm Hofe. Solving rostering tasks as constraint optimization. In *Selected Papers from the 3rd international conference on Practice and Theory of Automated Timetabling (PATAT)*, Springer-Verlag Lecture Notes on Computer Science, pages 280–297. 2000.
16. S Minton, M D Johnston, A B Philips, and P Laird. Minimizing conlicts: A heuristic repair method for constraint-satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.
17. K Miyashita, K Sycara, and R Mizoguchi. Modeling ill-structured optimization tasks through cases. *Decision Support Systems*, 17(4):345–364, 1996.

18. S Petrovic, G R Beddoe, and G Vanden Berghe. Case-based reasoning in employee rostering: learning repair strategies from domain experts. Technical Report NOTTCS-TR-2002-4, Automated Scheduling Optimisation and Planning Research Group, School of Computer Science and Information Technology, University of Nottingham, 2002.
19. S Petrovic, G R Beddoe, and G Vanden Berghe. Storing and adapting repair experiences in employee rostering. Technical Report NOTTCS-TR-2002-5, Automated Scheduling Optimisation and Planning Research Group, School of Computer Science and Information Technology, University of Nottingham, 2002.
20. W P Pierskalla and G J Rath. Nurse scheduling using mathematical programming. *Operations Research*, 24(5):857–870, 1976.
21. G Schmidt. Case-based reasoning for production scheduling. *International Journal of Production Economics*, 56-57:537–546, 1998.
22. S Scott and R Simpson. Case-bases incorporating scheduling constraint dimensions - experiences in nurse rostering. In *Advances in Case-Based Reasoning - EWCBR98*, Lecture Notes in Artificial Intelligence. Springer Verlag.
23. M Warner. Scheduling nursing personnel according to nurse preference: A mathematical programming approach. *Operations Research*, 24:842–856, 1976.