A Hybrid Evolutionary Approach to the Nurse Rostering Problem

Ruibin Bai, Edmund K. Burke, Graham Kendall, Jingpeng Li, and Barry McCollum

***Abstract*—Nurse rostering is an important search problem with many constraints. In the literature, a number of approaches have been investigated including penalty function methods to tackle these constraints within genetic algorithm frameworks. In this paper, we investigate an extension of a previously proposed stochastic ranking method, which has demonstrated superior performance to other constraint handling techniques when tested against a set of constrained optimization benchmark problems. An initial experiment on nurse rostering problems demonstrates that the stochastic ranking method is better at finding feasible solutions, but fails to obtain good results with regard to the objective function. To improve the performance of the algorithm, we hybridize it with a recently proposed simulated annealing hyper-heuristic (SAHH) within a local search and genetic algorithm framework. Computational results show that the hybrid algorithm performs better than both the genetic algorithm with stochastic ranking and the SAHH alone. The hybrid algorithm also outperforms the methods in the literature which have the previously best known results.**

***Index Terms*—Constrained optimization, constraint handling, evolutionary algorithm, local search, nurse rostering, simulated annealing hyper-heuristics.**

I. Introduction

NURSE ROSTERING is an important personnel scheduling problem that is faced by many large hospitals across the world. The problem involves producing daily schedules for nurses over a given time horizon. The objectives are to improve the hospitals' efficiency, to balance the workload among nurses and, more importantly, to satisfy various hard constraints, and as many soft constraints as possible, such as minimal nurse demands, "day-off" requests, personal preferences, etc. Depending on the practical situations and requirements in different hospitals, the type and number of constraints can be varied. Due to these constraints, the solution search space of nurse rostering problems is highly constrained with the feasible regions usually being disconnected. Although considerable research has been carried out in this area with many approaches effectively proposed, most standard methods have difficulties in dealing with these constraints. For example, as will be discussed in Section II-A, both the genetic algorithm in [2] and its improved version [1] are not able to consistently find feasible solutions for some problem instances.

In this paper, we aim to: 1) improve the constraint handling ability of a standard evolutionary approach by utilizing a stochastic ranking method. Stochastic ranking [3] is an effective constraint handling technique that has shown impressive performance over a set of constrained optimization benchmark problems. This method is shown to perform well for dealing with the difficult constraints in nurse rostering problems; 2) enhance the performance of the evolutionary method by hybridizing it with a simulated annealing hyper-heuristic (SAHH). It is well accepted that genetic algorithms are capable of searching large search spaces but are less effective in identifying local optima [4]; and 3) utilize a revised version of an emerging hyper-heuristic technique to enhance the performance of the algorithm. The hybrid algorithm is, in fact, very flexible and can be readily adapted to many other constrained optimization problems.

The remainder of this paper is structured as follows. Section II presents the nurse rostering problem that is addressed in this paper, followed by a brief overview of the related work for the problem. Section III reviews several constraint handling methods and specifically describes the stochastic ranking method that will be used in this paper. Section IV presents the initial experiments of the stochastic ranking genetic algorithm for the nurse rostering problem. In Section V, the proposed algorithm is enhanced by a simple version of a recently proposed SAHH [5]. Section VI concludes the paper.

II. The Nurse Rostering Problem

We will present a very brief overview of nurse rostering problems. A more comprehensive view can be found by consulting [6]–[8]. Research for nurse scheduling problems dates back to the early 1960s [9]–[13] where relatively simple mathematical models were proposed to minimize the cost of nurse recruitment in order to perform various tasks. Although these approaches are able to solve some nurse rostering problems, its performance greatly depends the size of the problem and the types of constraints to be handled [14]. With

Manuscript received June 13, 2007; revised December 23, 2008. Date of current version July 30, 2010. This work was supported by the U.K. Engineering and Physical Sciences Research Council under Grant Platform GR-S70197-01.

R. Bai is with the Division of Computer Science, University of Nottingham Ningbo, Ningbo 315100, China (e-mail: ruibin.bai@nottingham.edu.cn).

E. K. Burke, G. Kendall, and J. Li are with the School of Computer Science, University of Nottingham, Nottingham NG8 1BB, U.K. (e-mail: ekb@cs.nott.ac.uk; gxk@cs.nott.ac.uk; jpl@cs.nott.ac.uk).

B. McCollum is with the Department of Computer Science, School of Electronics, Electrical Engineering, and Computer Science, Queen's University Belfast, Belfast BT7 1NN, U.K. (e-mail: b.mccollum@qub.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2009.2033583

the advances in modern search and optimization techniques, a great deal of research has been carried out, in the last decade in particular, in the area of heuristic, metaheuristic and evolutionary personnel scheduling and nurse rostering [6]. Dowsland [15] proposed a multistage tabu search algorithm with the aid of several "chain-moves." Due to the highly constrained search space, the algorithm repeatedly switches between feasible and infeasible regions of the search space so that the search can transfer between different feasible regions even when they are disconnected. Experimental results have shown that this algorithm is able to find good quality solutions on the test problem instances. However, the tabu search algorithm relies highly on several specially designed "chain-moves." The performance of the algorithm may be not as good when tackling other problem instances with different search spaces. Burke *et al.* [16] employed a hybrid tabu search algorithm to solve a nurse rostering problem in Belgian hospitals. Apart from taking into account the common constraints, such as nurse demands for different categories, shift preferences, days off, etc., they also considered the constraints which arise due to the schedule in the previous schedule horizon. To tackle the highly constrained search space for which tabu search alone does not perform well, a hybrid method was proposed which hybridizes general tabu search with some heuristic search strategies. In [17], the nurse rostering problem was formulated into a multicriteria model so that users have more control and flexibility in adapting to actual situations in their hospitals. A new practical model for nurse rostering problems was recently proposed in [18] which introduces "time interval" personnel demands, a more flexible solution representation as opposed to shifts in most other models.

Burke *et al.* [19] applied a tabu search hyper-heuristic algorithm for nurse rostering problems. This algorithm is a flexible and generic framework which uses very little domain-specific information but can adapt to different problems by strategically choosing appropriate low-level heuristics. Beddoe and Petrovic [20] developed a case-based reasoning system and tested it on a real world nurse rostering problem. The system keeps a database of "cases" of previous constraint violations and the corresponding successful repair operations. A new problem can be solved by the approach that is retrieved by matching the current violation features with cases stored in the database. A genetic algorithm was used to select and combine a subset of features in case retrieval.

Aickelin and Dowsland [2] applied a genetic algorithm coupled with some problem-dependent genetic operators and local search heuristics. An enhanced version of the genetic algorithm was proposed in [1] which utilized a different solution encoding/decoding scheme and some specialized genetic operators in an "indirect genetic algorithm" framework. In both approaches, a carefully designed penalty function method was used to resolve the hard constraints. Due to the highly constrained search space of the nurse rostering problem, both genetic algorithms struggle to obtain feasible solutions for some of instances, although the second genetic algorithm performs slightly better than the genetic algorithm in [2]. Burke *et al.* [21] compared a memetic algorithm with a tabu search algorithm for nurse rostering problems and their computational results show that the memetic algorithm is able to obtain better quality solutions than both the genetic algorithm and a previously proposed tabu search approach in [16] provided that longer computational times are used. [22] and [23] are recent works on the nurse rostering problem which used Bayesian learning to combine several scheduling rules. Better results have been reported when compared with the genetic algorithms in [1], [2].

A. The Problem

In this paper, we address a real nurse rostering problem faced by a large U.K. hospital, originally studied in [15] and [2]. The problem is represented here for completeness. The formulation employed in those two studies represents a generic nurse rostering problem and has been used in several other studies. The problem is to make weekly schedules for about 30 nurses. Each days' schedule consists of a day shift and a night shift, and for each shift a feasible solution has to assign sufficient nurses to cover the actual demands which are subject to changes throughout the week. Two practical constraints have made this problem particularly challenging. Firstly, nurses have three different grades. A higher grade nurse can cover the demand for a lower grade nurse but not vice versa. Secondly, there are some part-time nurses who can only work a certain number of hours each week and may also not be able to work on certain shifts. The schedule should also be able to satisfy "day-off" requests by nurses. It should also spread some unpopular shifts (e.g., night and weekend shifts) among nurses for fairness. Dowsland [15] formulated this problem as an integer programming model. In her model, each nurse works on one of a number of predefined "shift patterns," which can be abstracted as a binary vector of length 14 (seven-day shifts and seven-night shifts). A value of one in the vector denotes a scheduled shift on for this nurse and zero a shift off. Each shift pattern of a nurse is associated with a penalty that represents its preferences. For completeness, we present the model here.

Given a number of, n, nurses with each nurse having a grade among the range $[1, g]$. Denote G_r the set of nurses with grades r or higher, R_{kr} the minimal demand of nurses of grade r for shift k, and F_i the set of feasible shift pattern for nurse i. Set $a_{jk} = 1$ if pattern k covers shift j and 0 otherwise. Let p_{ij} be the penalty cost of nurse i working on pattern j and the decision variables x_{ij} be

$$x_{ij} = \begin{cases} 1, & \text{nurse } i \text{ works on pattern } j \\ 0, & \text{otherwise.} \end{cases}$$

The objective is to minimize the following cost function

$$\min \quad f = \sum_{i=1}^{n} \sum_{j \in F_i} p_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in F_i} x_{ij} = 1 \tag{2}$$

$$\sum_{i \in G_r} \sum_{j \in F_i} a_{jk} x_{ij} \geq R_{kr} \ \forall k, r. \tag{3}$$

Constraint (2) ensures that each nurse works on exactly one specific shift pattern and constraint (3) makes sure that

there are sufficient nurses to cover each shift at each grade. Several methods have been used to tackle the constraint (3) which makes the search space highly constrained. Although a two-stage strategy and a penalty function method have respectively been used in [15] and [1], [2] in order to tackle the constraints, the proposed approaches either struggle to find feasible solutions or have to rely heavily upon problem-specific information. In this paper, we propose to tackle the constraints by using a generic stochastic ranking method which was shown to be very successful when solving 13 constrained optimization benchmark problems [3].

III. Evolutionary Algorithm and Constraint Handling Using Stochastic Ranking

Evolutionary algorithms are search techniques inspired from the natural evolution and selection principle of "survival of the fittest." For an optimization problem, a solution (individual) is usually encoded in a specially designed string (chromosome). A population of individuals is maintained and evolves from one generation to another through some genetic operations (i.e., crossover, mutation) and a selection method until some stopping criteria are met [24], [25]. Constraint handling is a common issue in many implementations of evolutionary algorithms. Depending on the problem, several techniques have been proposed in the literature. For example, Falkenauer [26] proposed a genetic algorithm with a specialized encoding schema and operators (crossover and mutation) for grouping problems so that the search only operates over the feasible solution space. A disadvantage of this approach is that not all the constraints can be handled by carefully designed encoding schemata and/or operators. In addition, the algorithm may not be efficient when the feasible regions of the solution space are disconnected. Another method is postreparation, which recovers the feasibility of the current solution if a constraint is violated after a crossover or mutation operation [27].

Penalty functions are among the most popular techniques and have been widely used in many applications [2], [19], [28]–[30]. The idea is to transform the constrained optimization problem into an unconstrained one by introducing a penalty term into the objective function to penalize constraint violations. Let X be the vector of decision variables and $f(X)$ be the original objective function. The transformed objective function $\phi(X)$ is often presented in the form of

$$\phi(X) = f(X) + \lambda\varphi(g_\pi(X);\, \pi \in \Pi) \tag{4}$$

where λ is the associated penalty coefficient and $\varphi(g_\pi(X))$ is a function that measures the severity of violations of the following constraints

$$g_\pi(X) \geq 0,\ \pi \in \Pi. \tag{5}$$

In the case of the nurse rostering problem addressed in this paper, the following function can be used to measure the violation of the covering constraints (3)

$$\varphi(g_\pi(X)) = \sum_{k=1}^{14}\sum_{r=1}^{g}\left\{\max\left\{0,\, R_{kr} - \sum_{i\in G_r}\sum_{j\in F_i} a_{jk}x_{ij}\right\}\right\}. \tag{6}$$

For convenience, we use ϕ and φ to denote $\phi(X)$ and $\varphi(g_\pi(X))$, respectively. Despite the popularity of the penalty function method, deciding on a proper value for penalty coefficient λ is challenging. In many cases, finding an optimal value for λ becomes a difficult optimization problem itself and is probably problem-dependent [3]. That is, parameter tuning is required for different problems (or even different problem instances). For example, in [1], [2], a similar form of penalty function is used to penalize violations of the covering constraint (3). The penalty coefficients are set after careful experimentation. Even so, both genetic algorithms in [1], [2] are struggling to find feasible solutions, especially for two of the 52 test instances that we study in this paper. For the two instances, these two genetic algorithms can only manage feasible solutions twice in 20 attempts. Adaptive approaches, where the value of λ is dynamically altered by the algorithm itself, are promising. The biggest advantage of these adaptive approaches is that constraints are handled by making use of some population information. Little domain-knowledge is required and there is no manual parameter tuning for λ [28], [29], [31]. Some other constraint handling methods rely on multiobjective optimization techniques [32]–[34], where constraints are treated as one or more objectives. For these methods, there is a problem of balancing the selection pressure between the objectives.

Another type of constraint handling method is stochastic ranking. It was initially proposed by Runarsson and Yao [3] as a technique to tackle constrained optimization problems in evolutionary algorithms. The underlying idea is to "fuzzify" the common ranking criteria by introducing a ranking probability P_f. The ranking can be obtained by a procedure similar to a stochastic version of the bubble-sort algorithm with N sweeps. In this method, the ranking is based on an objective function only if all the individuals are feasible. Otherwise, the ranking is stochastic. Denote by P_w the probability of an individual winning a comparison with an adjacent individual. It can be calculated by (see [3])

$$P_w = P_{fw}P_f + P_{\varphi w}(1 - P_f) \tag{7}$$

where P_{fw} and $P_{\varphi w}$ are respectively the probability of the individual winning according to the objective function and the penalty function. According to [3], the probability of an individual winning a comparison among S individuals is dependent on both the number of sweeps N and P_f. By fixing the number of sweeps N and by adjusting the probability P_f, we can balance the dominance of the objective function f and the penalty function φ [3]. In this research, we fix the number of sweeps $N = S$. When $P_f < 0.5$ the ranking is mainly dominated by the objective function f and when $P_f > 0.5$, the ranking favors smaller penalty function values φ. Since the ultimate purpose is to search for the best feasible solution, normally the parameter should be set where $P_f < 0.5$.

IV. Initial Experiments

An initial experiment was carried out to investigate the performance of the stochastic ranking method in comparison with the penalty function method in [1], [2] in the framework of a genetic algorithm. The solution is encoded as a

TABLE I
PARAMETERS FOR THE GENETIC ALGORITHM

Parameters	Settings
Population size	$ps = 1000$
Crossover	Simple one point crossover
Mutation	Change the shift pattern of a randomly selected nurse to a random but feasible pattern
Crossover rate	0.75
Mutation rate	0.02
Stop criteria	$gen' = 30$ continuous nonimprovement generations or the optimal solution is reached
P_f	0.25
Selection	Tournament selection with stochastic ranking (S=7) + elitism

vector of length n (i.e., number of nurses) with the position of each allele representing a nurse and its value the shift pattern index. This representation can automatically handle constraint (2). However, the covering constraint (3) will be handled using the stochastic ranking method. The parameter settings of the genetic algorithm are given in Table I. In order for a valid and sound comparison, all these parameter settings are the same as those used in [2] except for the selection strategy[1] which is based on stochastic ranking and elitism (i.e., the best solution always survives to the next generation) while in [2] the best 10% of solutions are directly copied to the next generation. In the same way as in [2], single point crossover is used. A mutation operator assigns a new random feasible shift pattern for a randomly selected nurse. Therefore, we did not tune these parameters when introducing the stochastic ranking method into the genetic algorithm.

Fig. 1(a) and (b) presents typical plots of the population feasibility level and transformed penalty cost against time.[2] Since the initial population is generated randomly, there is rarely a feasible solution at the beginning. As the search progresses, the feasibility level tends to increase and stabilizes at between 20% and 30% of the population. There are still a relatively large percentage of infeasible solutions in the population, probably due to the fact that we are dealing with a highly constrained search space. A large number of genetic operations (crossovers and mutations) would generate infeasible solutions. However, from Fig. 1(b) it can be seen that although the population feasibility level maintains a relatively stable value after 150 generations, the average penalty cost keeps reducing gradually over time, indicating that the overall solution quality of the population improves slowly over time. Note that maintaining a proportion of infeasible solutions in the population is useful for the search transferring between different disconnected feasible regions.

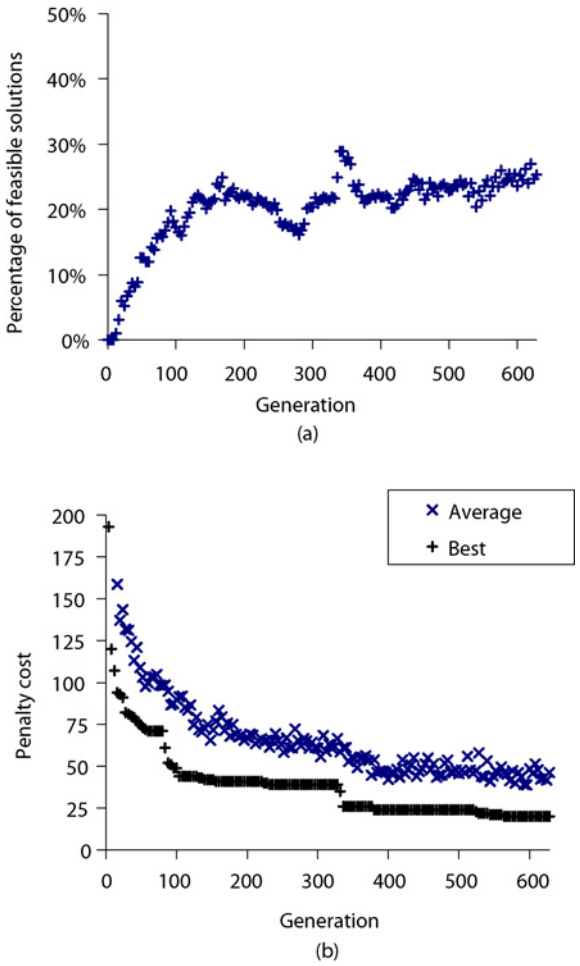

Fig. 1. Analyses of the population evolution of the stochastic ranking genetic algorithm. (a) Dynamics of the percentage of feasible solutions in the population. (b) Average and best penalty cost of the population over time.

Fig. 2(a) and (b) presents the results of the stochastic ranking genetic algorithm (SRGA) in comparison with the indirect genetic algorithm (IGA)[3] in [1] among 20 independent runs. Due to space limitation, we do not compare with the results in [2] but they are inferior to those in [1] both in terms of feasibility and objective values. Fig. 2(a) illustrates the advantages of the stochastic ranking method over the penalty function method. SRGA is able to find a feasible solution in all 20 runs for each of 52 instances. However, IGA is struggling for six instances, especially for the instances 49 and 50 where only 2 out of 20 attempts successfully find a feasible solution for the problem. Unfortunately, although it is able to find feasible solutions very quickly, the solution quality in terms of penalty costs is not as good as for IGA [see Fig. 2(b)]. In fact, the better quality solutions by IGA are mainly attributed to its special encoding schema and making use of some problem-specific information. To solve this problem, we hybridize the SRGA with a recently proposed SAHH algorithm [5].

[1]Note that duplicate solutions are not allowed in the population.

[2]The same transformed penalty cost function as in [1] was used for this analysis only.

[3]An arbitrary objective value of 200 is assigned to an infeasible solution.

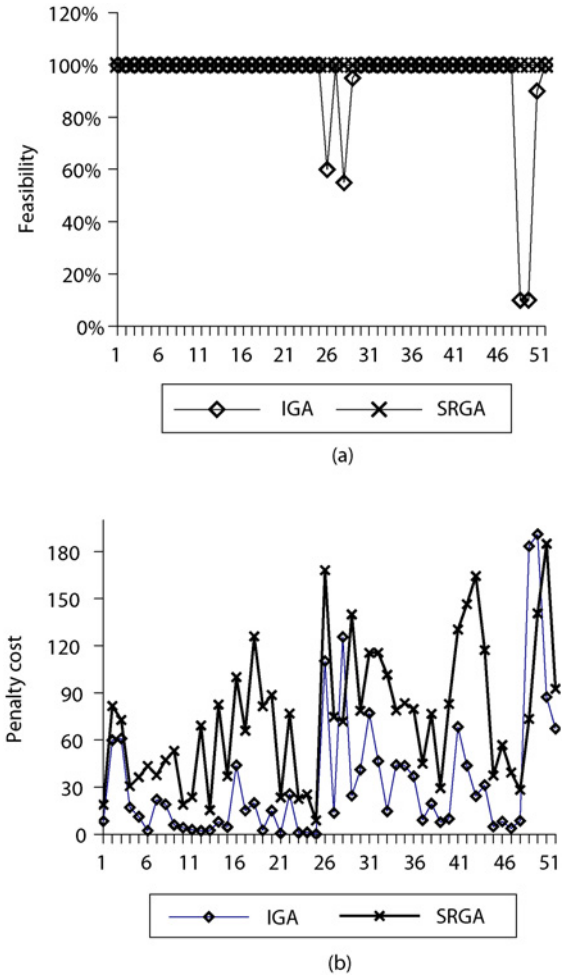

Fig. 2. Comparison of SRGA with IGA by [1]. (a) Percentages of feasible solutions obtained among 20 runs. (b) Average objective value over 20 runs.

V. Hybridization with an SAHH

A. The Hybrid Algorithm

Considerable research has shown that the performance of genetic algorithms can be improved by combining them with local search procedures. They are often referred to as *memetic algorithms* [4], [35]. In this paper, we hybridize the genetic algorithm with an SAHH approach that has demonstrated impressive performance over three difficult optimization problems [5]. Hyper-heuristics are high-level strategies that "choose heuristics to solve a given problem instance or search scenario" [36], [37]. A two-layer structure (separated by a domain barrier) can be adopted in order to increase the level of algorithmic independence over the problem domain. Two key components in the hyper-heuristic layer are the *heuristic selection mechanism* and the *simulated annealing acceptance criterion*. The heuristic selection mechanism strategically chooses between heuristics in order to adapt to different problem search scenarios. However, the simulated annealing acceptance criterion component, whose temperature is systematically changed during the search, ensures that only heuristic moves that have satisfied the criterion are accepted. Meanwhile, the heuristic selection component periodically monitors the performance of each heuristic and their acceptance ratios as feedback information to adapt its selection strategy to the current problem search scenarios. See [5] for more details.

However, it does not make sense to simply implement the entire SAHH algorithm into the genetic algorithm. Firstly, it is computationally expensive to execute an SAHH at each local search phase. Secondly, the main aim of a local search procedure in a memetic algorithm is to quickly identify local optima which the standard genetic algorithm finds difficult to locate. Therefore, there is no point in starting every local search with a high-temperature. The pseudocode for the proposed algorithm is outlined in Fig. 3.

The proposed algorithm was implemented in C++ and run on a PC with Intel Core 2 Duo 1.8 GHz CPU and 1 GB RAM. The parameters of the genetic algorithm remain the same as before except that the population size is decreased to 100 for computational considerations and that the stopping criterion is set to 60 s CPU time. The parameters with regard to the SAHHs are set as follows. $K = 20$, $t_s = 10.0$, $t_f = 0.5$, $nrep = 15$, $\beta = (t_s - t_f)/(nrep \cdot t_s \cdot t_f)$, based on some preliminary experiments. The temperature is decreased nonlinearly according to $t = t/(1 + \beta t)$ until $t < t_f$, at which point the temperature is reset to t_s. For the purpose of reducing computational time, the local search procedure, LS_SAHH, is a simplified version of the SAHH where the low-level heuristics are selected uniformly [see Fig. 4]. We also carried out some experiments on a hybridization of the genetic algorithm with an improvement-only local search approach (i.e., without the simulated annealing acceptance criterion). However, the results were not competitive with those presented here. Since the aim of the local search in the hybrid algorithm is to efficiently search for feasible local optima, the procedure only accepts feasible solutions, or in the case of equal infeasibility between the current solution and neighboring solution, the new solution is accepted according to the simulated annealing acceptance criteria. A total of nine simple low-level heuristics were used, drawn from [19]. For completeness, they are described here.

- H_1 Change the shift-pattern of a random nurse to another random feasible shift-pattern.
- H_2 Similar to H_1, except the acceptance criteria is "1st improving φ value."
- H_3 Same as H_1 but "1st improving φ and not deteriorating f."
- H_4 Same as H_1 but "1st improving f."
- H_5 Same as H_1 but "1st improving f and not deteriorating φ."
- H_6 Switch the shift-pattern type (i.e., from day to night and vice versa) of a random nurse if the solution is unbalanced.
- H_7 This heuristic tries to generate a balanced solution by switching the shift-pattern type [i.e., change a day shift-pattern with a night one if night shift(s) is unbalanced and vice versa. If both days and nights are not balanced, swap the shift patterns of two nurses who are working on different shift-pattern types].
- H_8 This heuristic tries to find the first move that improves f by changing the shift pattern of a random

step1: Initialization: generate an initial population, set start temperature t_s, stopping temperature t_f and temperature reduction rate β, $t = t_s$. Set the number of iterations for local search K.
step2: Apply genetic operations (crossover and mutation).
step3: Stochastic ranking and selection.
step4: For each individual I_u, call $LS\_SAHH(I_u, K)$.
step5: Update temperature: if $(t > t_f)$ then $t = t/(1 + \beta t)$, otherwise $t = t_s$
step6: Goto step2.

Fig. 3. Pseudocode of the hybrid algorithm.

Input: I_u, K, low-level heuristics H_i, i=1,...,m.
for i=1 to K
  Select a heuristic H_i uniformly.
  Sample a new solution I'_u from I_u using heuristic H_i
  if $(\varphi(g_\pi(I'_u)) < \varphi(g_\pi(I_u)))$ then $I_u \leftarrow I'_u$.
  else if $(\varphi(g_\pi(I'_u)) = \varphi(g_\pi(I_u)))$ then
    Calculate the difference in objective function $\delta = f(I'_u) - f(I_u)$.
    if ($\delta \leq 0$ or $e^{-\delta/t} > Rand(0, 1)$) then $I_u \leftarrow I'_u$.
  endif
endfor
Output I_u

Fig. 4. Procedure $LS\_SAHH(I_u, K)$.

nurse and assign the abandoned shift pattern to another nurse.

H_9 Same as H8 but "1st improving f without worsening φ."

B. Comparison with Other Approaches

The proposed hybrid algorithm was applied to the same 52 instances as in [2], with each instance being solved 20 times using independent random seeds. The detailed results of the hybrid algorithm are presented in Table III and Fig. 7. We now make comparisons with the tabu search hyper-heuristic (TSHH) [19], the IGA [1], and a more recently proposed estimation of distribution algorithm (EDA) [23]. Since the TSHH was run on a slower computer, for the purpose of a fair comparison, we re-run the original TSHH program provided by the corresponding authors on the same PC used for our hybrid algorithm with the same computational time limit (i.e., 60 s). With this extra computational power, TSHH is able to improve its average results for 37 out 52 instances, compared with its best previous results under parameter configuration HH1:4L in [38]. The proposed algorithm outperforms a recently proposed evolutionary algorithm on average [39]. Detailed results are not included in this paper since it has only recently appeared.

Table II presents a comparison of the average objective values by the proposed hybrid algorithm and those by re-running the TSHH. The results are based on 20 independent runs both for the TSHH and the hybrid algorithm. It can be seen that for 19 instances both algorithms obtained the same results. The hybrid algorithm performed better than TSHH for 23 instances while TSHH produced slight better results for the other ten instances. In terms of overall average objective values across 52 instances, the hybrid algorithm performed better than TSHH. To make a further comparison, the standard nonparametric signed rank test was carried out for each of the 33 instances for which TSHH and the hybrid algorithm performed differently. However, a further 11 instances were excluded for the significance test since the results from the both algorithms are every similar (i.e., the number of nonzero difference in samples is less than five). The signed rank test results for the remaining 22 instances show that the hybrid algorithm significantly performs better than the TSHH for 20 instances and for two instances, there is no significant difference between them.

The detailed results of IGA and EDA are presented in Figs. 5 and 6, respectively. Comparisons are made in three aspects: **#inf.** is the number of unsuccessful runs (out of 20 total independent runs) that have failed to find a feasible solution by the given algorithm. **#opt.** denotes the number of successful attempts that have found an optimal solution and **# within** 3 is the number of runs that found a solution within 3 penalty costs away from the optimum. These solutions are considered to be of good quality. The optimal solutions were obtained by a standard Integer Programming (IP) package which may be impractical due to the high financial costs [1].

It can be seen that both the EDA and the proposed hybrid algorithms perform better than the IGA in terms of finding feasible solutions. The IGA has difficulties in finding feasible solutions for six problem instances while both the EDA and the hybrid algorithm can find feasible solutions in all 20 runs for all the instances. In general, the performance of the proposed hybrid algorithm is much better than both IGA and EDA. Among 20 runs, the hybrid algorithm can solve all the instances to optimality except the second instance. For 49 out of 52 instances, the hybrid algorithm obtained a good quality

TABLE II
HYBRID ALGORITHM VERSUS THE TABU SEARCH HYPER-HEURISTIC

Set	1	2	3	4	5	6	7	8	9	10	11	12	13
TSHH	**8.0**	**50.1**	**50.0**	**17.0**	**11.0**	**2.0**	11.2	15.1	**3.0**	**2.0**	**2.0**	**2.0**	**2.0**
Hybrid Algorithm	**8.0**	50.3	**50.0**	**17.0**	**11.0**	2.1	**11.1**	**14.1**	3.1	2.2	**2.0**	**2.0**	**2.0**
Set	14	15	16	17	18	19	20	21	22	23	24	25	26
TSHH	**3.0**	**3.0**	38.0	11.2	18.7	**1.0**	7.2	**0.0**	**25.0**	**0.0**	**1.0**	**0.0**	**48.0**
Hybrid Algorithm	**3.0**	**3.0**	**37.0**	**9.0**	**18.0**	1.1	**7.0**	**0.0**	25.1	0.1	**1.0**	**0.0**	**48.0**
Set	27	28	29	30	31	32	33	34	35	36	37	38	39
TSHH	2.7	63.4	**15.0**	**35.0**	66.9	40.1	**10.3**	39.8	36.7	33.2	**5.0**	**13.0**	**5.0**
Hybrid Algorithm	**2.0**	**63.0**	15.2	**35.0**	**65.0**	**40.0**	10.4	**38.0**	**35.0**	**32.0**	**5.0**	13.1	**5.0**
Set	40	41	42	43	44	45	46	47	48	49	50	51	52
TSHH	7.8	55.5	39.9	23.2	25.4	**3.0**	4.7	**3.0**	4.6	28.2	107.9	**74.0**	61.1
Hybrid Algorithm	**7.0**	**54.0**	**38.0**	**22.0**	**19.6**	**3.0**	**4.0**	**3.0**	**4.0**	**27.0**	**107.0**	**74.0**	**58.0**
Overall Average	TSHH:22.8, Hybrid Algorithm: 21.3												

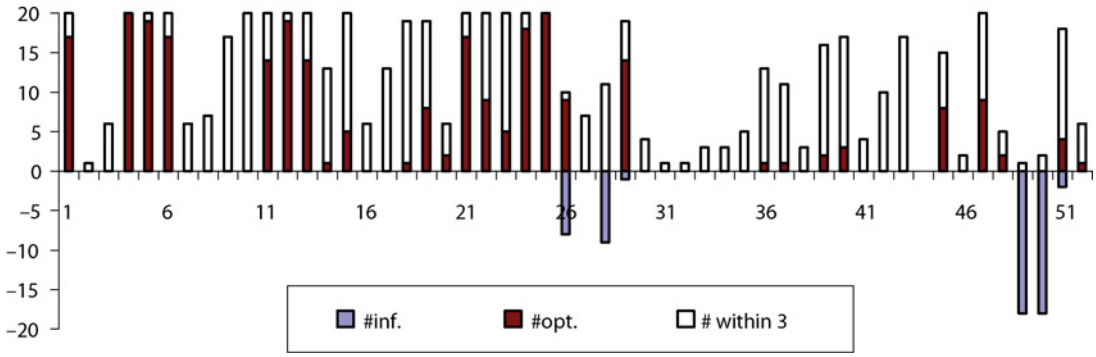

Fig. 5. Detdetailed results by IGA.

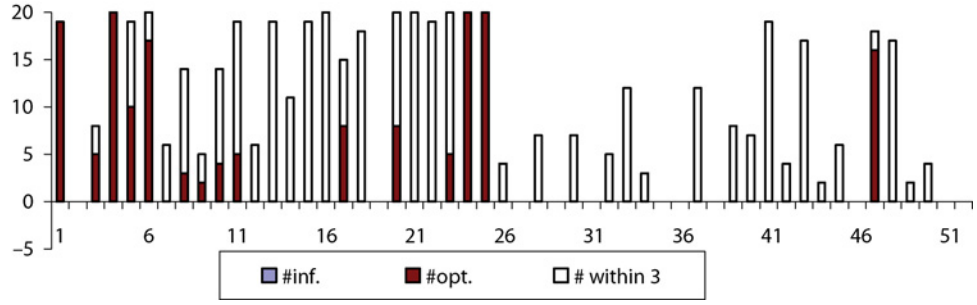

Fig. 6. Detailed results by EDA with "ant-miner" heuristics.

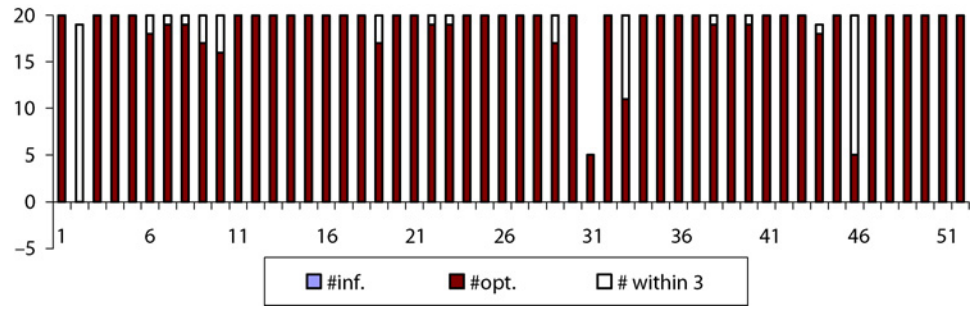

Fig. 7. Detailed results by the hybrid algorithm.

TABLE III
HYBRID ALGORITHM VERSUS SAHH

Set	IP	SAHH				Hybrid Algorithm			
		Best	Mean	Worst	Stdev	Best	Mean	Worst	Stdev
1	8	8	8.0	8	0.00	8	8.0	8	0.00
2	49	49	50.9	55	2.13	50	50.3	56	1.34
3	50	50	50.0	50	0.00	50	50.0	50	0.00
4	17	17	17.0	17	0.00	17	17.0	17	0.00
5	11	11	11.0	11	0.00	11	11.0	11	0.00
6	2	2	2.0	2	0.00	2	2.1	3	0.31
7	11	11	11.0	11	0.00	11	11.1	12	0.22
8	14	14	14.1	15	0.31	14	14.1	15	0.22
9	3	3	3.0	3	0.00	3	3.2	4	0.37
10	2	2	2.5	4	0.69	2	2.2	3	0.41
11	2	2	2.0	2	0.00	2	2.0	2	0.00
12	2	2	2.0	2	0.00	2	2.0	2	0.00
13	2	2	2.0	2	0.00	2	2.0	2	0.00
14	3	3	3.3	4	0.44	3	3.0	3	0.00
15	3	3	3.0	3	0.00	3	3.0	3	0.00
16	37	37	41.4	66	8.66	37	37.0	37	0.00
17	9	9	10.1	15	1.67	9	9.0	9	0.00
18	18	18	18.7	28	2.25	18	18.0	18	0.00
19	1	1	1.3	4	0.72	1	1.2	2	0.37
20	7	7	8.3	21	3.18	7	7.0	7	0.00
21	0	0	0.2	1	0.41	0	0.0	0	0.00
22	25	25	25.3	27	0.57	25	25.1	26	0.22
23	0	0	0.2	1	0.41	0	0.1	1	0.22
24	1	1	1.0	1	0.00	1	1.0	1	0.00
25	0	0	0.2	1	0.41	0	0.0	0	0.00
26	48	48	91.1	198	54.04	48	48.0	48	0.00
27	2	2	4.4	13	4.49	2	2.0	2	0.00
28	63	63	63.3	65	0.55	63	63.0	63	0.00
29	15	15	15.3	18	0.72	15	15.2	16	0.37
30	35	35	35.7	40	1.42	35	35.0	35	0.00
31	62	62	64.7	66	1.84	62	65.0	66	1.76
32	40	40	40.1	41	0.22	40	40.0	40	0.00
33	10	10	15.6	103	20.64	10	10.5	11	0.51
34	38	38	38.1	39	0.22	38	38.0	38	0.00
35	35	35	35.9	39	1.23	35	35.0	35	0.00
36	32	32	32.7	33	0.49	32	32.0	32	0.00
37	5	5	5.0	5	0.00	5	5.0	5	0.00
38	13	13	13.1	15	0.45	13	13.1	15	0.45
39	5	5	5.0	5	0.00	5	5.0	5	0.00
40	7	7	7.5	9	0.69	7	7.1	8	0.22
41	54	54	61.7	83	12.11	54	54.0	54	0.00
42	38	38	38.8	40	0.55	38	38.0	38	0.00
43	22	22	23.2	32	3.04	22	22.0	22	0.00
44	19	19	27.3	34	5.21	19	19.6	29	2.24
45	3	3	3.4	5	0.75	3	3.0	3	0.00
46	3	3	4.8	6	0.83	3	4.0	5	0.69
47	3	3	3.0	3	0.00	3	3.0	3	0.00
48	4	4	4.3	5	0.44	4	4.0	4	0.00
49	27	27	31.9	118	20.28	27	27.0	27	0.00
50	107	107	107.9	109	0.81	107	107.0	107	0.00
51	74	74	74.1	75	0.31	74	74.0	74	0.00
52	58	58	59.3	73	3.92	58	58.0	58	0.00
Av	21.1	21.1	23.0	31.3	3.02	21.2	21.3	21.8	0.19

solution (i.e., solutions within three penalty cost away from optimality) on each of 20 independent runs.

In terms of computational time, our hybrid algorithm was run on a PC with Intel Core 2 Duo 1.8 GHz CPU and 1 GB RAM but the program is a sequential one (i.e., no advantage is gained here by using a PC with dual CPU). The stopping criterion is 60 s computational time per run per instance although an optimal solution could be found well before this time limit for the majority of the instances. Most of the computational time was consumed due to the local search phase SAHH_LS, which is computationally expensive. IGA is very fast, with an average time of 9.3 s per run per instance on a Pentium II PC but the results for many instances are not competitive. EDA takes much longer computational time (2–3 min on average) on a Pentium IV 2.0 GHz PC with 512 MB RAM, which has a similar speed to our experiment PC.

Table III presents more detailed results of the proposed hybrid algorithm in comparison with the SAHH in [5].[4]

[4]The SAHH was re-run on the experiment PC used in this paper with the same computation time, however no noticeable improvement was observed.

Again, the signed rank test was carried out to test the significance difference between them. For 28 instances, both algorithms performed similarly with less than five nonzero differences among each sampled results. Therefore, they were not considered for the significance test. For the remaining 24 instances, test results indicate that the hybrid algorithm performs better than SAHH for 19 instances and there is no significant difference for the other five instances.

Judged from the tables, the hybrid algorithm seems to be more consistent in producing good quality solutions. Take instance 16 for example, both the average penalty costs and standard deviation by SAHH are much larger than the hybrid algorithm's. The search space of this instance seems to have many disconnected feasible regions. As a population-based approach, the proposed hybrid algorithm is capable of searching in a larger search space than SAHH, which is a single point search method. Similar results can be observed for instances 18, 20, 26, 33, 41, and 49. Over the 52 instances, the average penalty costs (respectively the worst penalty cost and standard deviation) by the hybrid algorithm have been reduced by 7.5% (respectively 30.3% and 93.7%) compared with SAHH.

Another advantage of the proposed hybrid approach is its simplicity of implementation and flexibility to be adapted to other constrained optimization problems. Compared with the penalty function method and other constraint handling techniques, stochastic ranking is simpler and more generic. It is not based on domain-specific structures and hence can be used for various constraint handling situations within an evolutionary algorithm framework. Meanwhile, the SAHH could complement the drawbacks of a conventional evolutionary algorithm with its better capability to capture local optima efficiently.

VI. Conclusion

This paper has considered a real-world nurse rostering problem which has a highly constrained search space. Several previous heuristic approaches have been proposed for this problem, which have either struggled to find feasible solutions or failed to produce high-quality solutions efficiently in terms of the objective function. In this paper, we proposed a hybrid algorithm for this problem which combines a genetic algorithm and an SAHH. In this algorithm, a stochastic ranking method was used to improve the constraint handling capability of the genetic algorithm while an SAHH procedure was incorporated in order to locate local optima more efficiently. Compared with genetic algorithms that use penalty function methods as a constraint handling approach, the stochastic ranking method have demonstrated better performance with regard to feasibility. To improve the solution quality in terms of the objective function, an SAHH algorithm was hybridized with the genetic algorithm. Experimental results on 52 problem instances has demonstrated the high-performance and consistency by this hybrid approach when compared with other approaches for this problem. The contribution of this paper is the presentation of a robust hybrid algorithm for the nurse rostering problem. The algorithm is also simple and flexible and provides significant potential for extension to other constrained optimization problems.

References

[1] U. Aickelin and K. A. Dowsland, "An indirect genetic algorithm for a nurse scheduling problem," *Comput. Oper. Res.*, vol. 31, no. 5, pp. 761–778, 2003.

[2] U. Aickelin and K. A. Dowsland, "Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem," *J. Scheduling*, vol. 3, no. 3, pp. 139–153, 2000.

[3] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 4, no. 3, pp. 284–294, Sep. 2000.

[4] W. E. Hart, N. Krasnogor, and J. Smith, Eds., *Recent Advances in Memetic Algorithms* (Studies in Fuzziness and Soft Computing Series 166), Berlin, Germany: Springer-Verlag, 2004, ch. 1, pp. 3–30.

[5] R. Bai, J. Blazewicz, E. K. Burke, G. Kendall, and B. McCollum, "A simulated annealing hyper-heuristic methodology for flexible decision support," School of CSiT, Univ. Nottingham, Nottingham, U.K., Tech. Rep. NOTTCS-TR-2007-8, 2007.

[6] E. K. Burke, P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem, "The state of the art of nurse rostering," *J. Scheduling*, vol. 7, no. 6, pp. 441–499, 2004.

[7] D. Sitompul and S. U. Randhawa, "Nurse scheduling models: A state-of-the-art review," *J. Soc. Health Syst.*, vol. 2, no. 1, pp. 62–72, 1990.

[8] R. Hung, "Hospital nurse scheduling," *J. Nurs. Adm.*, vol. 25, nos. 7–8, pp. 21–23, 1995.

[9] H. Wolfe and J. P. Young, "Staffing the nursing unit: Part I," *Nurs. Res.*, vol. 14, no. 3, pp. 236–243, 1965.

[10] H. Wolfe and J. P. Young, "Staffing the nursing unit: Part II," *Nurs. Res.*, vol. 14, no. 4, pp. 199–303, 1965.

[11] J. P. Howell, "Cyclical scheduling of nursing personnel," *Hospitals*, vol. 40, no. 2, pp. 77–85, 1966.

[12] D. W. Warner, "Scheduling nurse personnel according to nursing preference: A mathematical programming approach," *Oper. Res.*, vol. 24, no. 5, pp. 842–856, 1976.

[13] H. E. Miller, W. P. Pierskalla, and G. Rath, "Nurse scheduling using mathematical programming," *Oper. Res.*, vol. 24, no. 5, pp. 857–870, 1976.

[14] K. A. Dowsland and J. M. Thompson, "Solving a nurse scheduling problem with knapsacks, networks and tabu search," *J. Oper. Res. Soc.*, vol. 51, no. 7, pp. 393–394, 2000.

[15] K. A. Dowsland, "Nurse scheduling with tabu search and strategic oscillation," *Eur. J. Oper. Res.*, vol. 106, nos. 2–3, pp. 393–407, 1998.

[16] E. K. Burke, P. De Causmaecker, and G. Vanden Berghe, "A hybrid tabu search algorithm for the nurse rostering problem," in *Proc. Simulated Evol. Learning: Selected Papers 2nd Asia-Pacific Conf. Simulated Evol. Learn. (SEAL)*, Lecture Notes in Computer Science Series 1585. Berlin, Germany: Springer-Verlag, 1999, pp. 187–194.

[17] E. K. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe, "A multicriteria meta-heuristic approach to nurse rostering," in *Proc. Practice Theory Automated Timetabling: Selected Revised Papers 3rd Practice Theory Automated Timetabling Int. Conf.*, Lecture Notes in Computer Science Series 2079. Berlin, Germany: Springer-Verlag, 2001, pp. 118–131.

[18] E. K. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe, "Metaheuristics for handling time interval coverage constraints in nurse scheduling," *Appl. Artif. Intell.*, vol. 20, no. 9, pp. 743–766, 2006.

[19] E. K. Burke, G. Kendall, and E. Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," *J. Heuristics*, vol. 9, no. 6, pp. 451–470, 2003.

[20] G. Beddoe and S. Petrovic, "Selecting and weighting features using a genetic algorithm in a case-based reasoning approach to personnel rostering," *Eur. J. Oper. Res.*, vol. 175, no. 2, pp. 649–671, 2006.

[21] E. Burke, P. Cowling, P. De Causmaecker, and G. V. Berghe, "A memetic approach to the nurse rostering problem," *Appl. Intell.*, vol. 15, no. 3, pp. 199–214, 2001.

[22] U. Aickelin and J. Li, "An estimation of distribution algorithm for nurse scheduling," *Ann. Oper. Res.*, vol. 155, no. 1, pp. 289–309, 2007.

[23] U. Aickelin, E. K. Burke, and J. Li, "An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering," *J. Oper. Res. Soc.*, vol. 58, no. 12, pp. 1574–1585, 2007.

[24] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: Univ. Michigan Press, 1992, ch. 2, pp. 20–31.

[25] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA: Addison Wesley, 1989, ch. 1, pp. 1–26.
[26] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *J. Heuristics*, vol. 2, no. 1, pp. 5–30, 1996.
[27] P. Chu and J. E. John, "A genetic algorithm for the multidimensional knapsack problem," *J. Heuristics*, vol. 4, no. 1, pp. 63–86, 1998.
[28] D. W. Coit, A. E. Smith, and D. M. Tate, "Adaptive penalty methods for genetic optimization of constrained combinatorial problems," *INFORMS J. Comput.*, vol. 8, no. 2, pp. 173–182, 1996.
[29] K. Deb, "An efficient constraint handling method for genetic algorithms," *Comput. Methods Appl. Mech. Eng.*, vol. 186, nos. 2–4, pp. 311–338, 2000.
[30] S. Venkatraman and G. G. Yen, "A generic framework for constrained optimization using genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 9, no. 4, pp. 424–435, Aug. 2005.
[31] R. Farmani and J. A. Wright, "Self-adaptive fitness formulation for constrained optimization," *IEEE Trans. Evol. Comput.*, vol. 7, no. 5, pp. 445–455, Oct. 2003.
[32] P. D. Surry and N. J. Radcliffe, "The comoga method: Constrained optimization by multiobjective genetic algorithms," *Control Cybern.*, vol. 26, no. 3, pp. 391–412, 1997.
[33] F. Y. Cheng and D. Li, "Multiobjective optimization design with pareto genetic algorithm," *J. Struct. Eng.*, vol. 123, no. 9, pp. 1252–1261, 1997.
[34] C. A. C. Coello, "Treating constraints as objectives for single-objective evolutionary optimization," *Eng. Optim.*, vol. 32, no. 3, pp. 275–308, 2000.
[35] N. Krasnogor and J. E. Smith, "A tutorial for competent memetic algorithms: Model, taxonomy and design issues," *IEEE Trans. Evol. Comput.*, vol. 9, no. 5, pp. 474–488, Oct. 2005.
[36] E. K. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Dordrecht, The Netherlands: Kluwer, 2003, pp. 457–474.
[37] P. Ross, "Hyper-heuristics," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, E. K. Burke and G. Kendall, Eds. New York; Springer-Verlag, 2005, ch. 17, pp. 529–556.
[38] E. Soubeiga, "Development and application of hyperheuristics to personnel scheduling," Ph.D. dissertation, School Comput. Sci., Univ. Nottingham, Nottingham, U.K., Jun. 2003.
[39] J. Li, U. Aickelin, and E. K. Burke, "A component-based heuristic search method with evolutionary eliminations for hospital personnel scheduling," *INFORMS J. Comput.*, vol. 21, no. 3, pp. 468–479, 2009.

Ruibin Bai received the Bachelor and Master degrees in mechanical engineering from Northwestern Polytechnic University, Shaanxi, China, in 1999 and 2002, respectively, and the Ph.D. degree in computer science from the University of Nottingham, Nottingham, U.K., in 2005.

Before taking a Lectureship at the Division of Computer Science, University of Nottingham, Ningbo, Ningbo, China, he spent two years with the School of Computer Science, University of Nottingham, as a Postdoctoral Research Fellow. His research interests include the investigation and development of adaptive and intelligent decision support systems for scheduling and timetabling, logistics, transportation, layout, and space planning.

Dr. Bai has been working collaboratively with several internationally established researchers from different countries, and has served as a Referee for several leading journals, such as IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *Omega*, *Journal of Heuristics*, *Annals*, and *Asia-Pacific Journal of Operational*, *Journal of Retailing*, etc. He has also refereed for a number of internationally recognized conferences, such as Genetic and Evolutionary Computation 2009, Evolutionary Computation in Combinatorial Optimization 2006–2009, Computational Intelligence in Scheduling 2007–2009, Intelligent Systems Design and Applications 2006/2007, Industrial and Engineering Applications of Artificial Intelligence and Expert Systems 2007, International Conference on Tools with Artificial Intelligence 2006, and Artificial Intelligence and Pattern Recognition 2007. He is a Member of the Automated Scheduling, Optimization, and Planning Research Group.

Edmund Burke received the B.Ed., M.S., and Ph.D. degrees from the University of Leeds, Leeds, U.K., in 1986, 1987, and 1991, respectively. He also received the FORS, FBCS, and CITP degrees.

He is the Dean of the Faculty of Science, the School of Computer Science, University of Nottingham, Nottingham, U.K., where he leads the Automated Scheduling, Optimization, and Planning Research Group in the School of Computer Science. He has edited or authored 14 books and has published over 180 refereed papers. His research interests are in computational search at the interface of Operational Research and Computer Science. In particular, he works in heuristic and metaheuristic design across a wide range of applications including timetabling, scheduling, cutting/packing, and bioinformatics.

Prof. Burke is a Member of the Engineering and Physical Sciences Research Council (EPSRC) Strategic Advisory Team for Mathematics, the U.K. Computing Research Committee, and the Editorial Board of Memetic Computing. He is a Fellow of the Operational Research Society and the British Computer Society. He is Editor-in-Chief of the *Journal of Scheduling*, an Area Editor (for Combinatorial Optimization) of the *Journal of Heuristics*, an Associate Editor of the *INFORMS Journal on Computing*, and an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He has played a leading role in the organization of several major international conferences in his research field in the last few years. He has been awarded 47 externally funded grants worth over £11 million from a variety of sources including the EPSRC, the Economic and Social Research Council, the Biotechnology and Biological Sciences Research Council, the European Union, the Research Council of Norway, the East Midlands Development Agency, the Higher Education Funding Council for England (HEFCE), the Teaching Company Directorate, and the Joint Information Systems Committee of the HEFCEs and the commercial organizations. This funding portfolio includes being the Principal Investigator on a recently awarded EPSRC Science and Innovation award of £2 million, an EPSRC grant of £2.6 million to investigate the automation of the heuristic design process, and an EPSRC platform grant worth £423 thousand.

Graham Kendall received the Bachelor's degree in computation (1st Class, Hons.) from the University of Manchester Institute of Science and Technology, Manchester, U.K. in 1997, and the Ph.D. degree in computer science from the University of Nottingham, Nottingham, U.K. He graduated in 2001.

His previous experience includes almost 20 years in the Information Technology industry where he held both technical and managerial positions. Currently, he is a Professor at the School of Computer Science, University of Nottingham, Nottingham, U.K. During his career, he has edited or authored ten books, published over 90 refereed papers, and has guest edited special issues of journals including *Annals of Operations Research*, the *Journal of Scheduling*, and the International Computer Games Association. His research interests include adaptive learning (with an emphasis on evolving games), heuristic development (particularly hyper-heuristics), optimization, scheduling (particularly sports), and artificial intelligence.

Prof. Kendall is a Member of the Automated Scheduling, Optimization, and Planning Research Group. He is a Fellow of the Operational Research Society. He is an Associate Editor of seven international journals, including two IEEE journals: IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND ARTIFICIAL INTELLIGENCE IN GAMES. He chaired the Multidisciplinary International Conference on Scheduling: Theory and Applications in 2003, 2005, 2007, 2009, and has chaired several other international conferences, which has included establishing the IEEE Symposium on Computational Intelligence and Games. He chaired this symposium in 2005 and 2006. He has been a Member of the program (or refereeing) committees of over 130 international conferences over the last few years. He has been awarded externally funded grants worth over £6 million from a variety of sources including Engineering and Physical Sciences Research Council (EPSRC) and commercial organizations. He is a Director of two spin out companies (EventMAP Ltd., Nottingham, U.K., and Aptia Solutions Ltd., Nottingham, U.K.).

Jingpeng Li received the B.S. degree in computer science from Hangzhou Dianzi University, Hangzhou, China, in 1990, the M.S. degree in computational mathematics from the Huazhong University of Science and Technology, Huazhong, China, in 1998, and the Ph.D. degree in computer science from the University of Leeds, Leeds, U.K., in 2002.

He joined the School of Informatics, University of Bradford, Bradford, U.K., as a Research Assistant in 2003. Since 2004, he has been with the School of Computer Science, University of Nottingham, Nottingham, U.K., and is currently a Permanent Research Fellow within the Automated Scheduling, Optimization, and Planning Research Group. In total, he has published 25 papers in a wide variety of the world's leading journals and conference proceedings. His research interests include artificial intelligence, evolutionary computation, data mining, and operations research.

Dr. Li has worked on five U.K. government-funded Engineering and Physical Sciences Research Council projects on topics including space allocation, next generation decision support, novel research directions in personnel rostering, general optimization systems, and human scheduling algorithms. He is acting as a Referee for a number of leading journals (such as the *Journal of the Operational Research Society* and the *European Journal of Operational Research*), and has served on the program committees for many international conferences (such as the 2009 Genetic and Evolutionary Computation Conference and the 2008 International Conference on Parallel Problem Solving from Nature).

Barry McCollum received the B.S. (Hons.) and M.S. degrees from the School of Electronics, Electrical Engineering and Computer Science, Queen's University, Belfast, U.K. in 1992 and 1994, respectively. After working in University Administration for three years at the University of Nottingham, Nottingham, U.K., he returned to Belfast to complete his PhD in applied artificial intelligence and graduated in 2000.

He is the Leader of the Scheduling and Optimization Team within the Knowledge and Data Engineering Research Group. He has published over 90 refereed articles on automated timetabling and has managed externally funded projects worth over £1.5 million from government and industrial sources. He has positioned his research agenda to exploit practical research and commercial knowledge to further the understanding of real world automated decision support with the goal of making a major impact on addressing the complexity and uncertainty of optimization issues as they exist within modern industrial settings. He has successfully combined practical and theoretical issues through his two spin out companies, Realtime Solutions (www.realtimesolutions-uk.com) and EventMAP Ltd., Nottingham, U.K. (www.eventmap-uk.com). As Managing Director of both companies, he strives to solve real world timetabling problems using leading edge technologies within a structured business context.

Dr. McCollum is the Permanent Industrial Chair of the Multidisciplinary International Conference on Scheduling: Theory and Applications conference series with particular emphasis on knowledge transfer between the industrial and scientific communities. He recently led the International Timetabling Competition (www.cs.qub.ac.uk/itc2007), which aimed to present the scientific community with more realistic problems. He acts on the steering committee of the Practice and Theory of Automated Timetabling series of conferences (http://www.asap.cs.nott.ac.uk/patat/patat-index.shtml), hosting PATAT 10 in Queen's University in 2010.