A Hybrid Evolutionary Approach to the Nurse Rostering Problem

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

***Abstract*— Nurse rostering is a difficult 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 optimisation benchmark problems. An initial experiment on nurse rostering problems demonstrates that the stochastic ranking method is better in finding feasible solutions but fails to obtain good results with regard to the objective function. To improve the performance of the algorithm, we hybridise it with a recently proposed simulated annealing hyper-heuristic within a local search and genetic algorithm framework. The hybrid algorithm shows significant improvement over both the genetic algorithm with stochastic ranking and the simulated annealing hyper-heuristic alone. The hybrid algorithm also considerably outperforms the methods in the literature which have the previously best known results.**

***Index Terms*— Nurse rostering, Evolutionary algorithm, Local search, Simulated annealing hyper-heuristics, Constrained optimisation, Constraint handling**

I. Introduction

Nurse rostering is a difficult and 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 2.1, both the genetic algorithm in [1] and its improved version [2] 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 utilising a stochastic ranking method. Stochastic ranking [3] is an effective constraint handling technique that has shown impressive performance over a set of constrained optimisation 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 hybridising it with a simulated annealing hyper-heuristic. It is well accepted that genetic algorithms are capable of searching large search spaces but are less effective in identifying local optima [4]. 3). We utilise 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 optimisation problems.

The remainder of this paper is structured as follows. Section 2 presents the nurse rostering problem that is addressed in this paper, followed by a brief overview of the related work for the problem. Section 3 reviews several constraint handling methods and specifically describes the stochastic ranking method that will be used in this paper. Section 4 presents the initial experiments of the stochastic ranking genetic algorithm for the nurse rostering problem. In section 5, the proposed algorithm is enhanced by a simple version of a recently proposed simulated annealing hyper-heuristic [5]. Section 6 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 1960's [9]–[13] where relatively simple mathematical models were proposed to minimise the cost of nurse recruitment in order to perform various tasks. Although these approaches are able to solve small sized problems, the computational time for large sizes problems is usually prohibitive for most practical applications [14].With the advances in modern search and optimisation techniques, a great deal of research have 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 multi-stage 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

Manuscript received xx xx, 2007. This work was supported by the UK EPSRC under Grant Platform (Ref:GR/S70197/01).

Ruibin Bai, Edmund K. Burke, Graham Kendall and Jingpeng Li are with the School of Computer Science & IT, University of Nottingham, Nottingham NG8 1BB, UK (email:rzb|ekb|gxk|jpl@cs.nott.ac).

Barry McCollum is with Department of Computer Science, Queen's University Belfast, Belfast BT7 1NN, UK (email: b.mccollum@qub.ac.uk).

solve a nurse rostering problem in Belgian hospitals. Apart from taking account into the common constraints, such as nurse demands for different categories, shift preferences, day-offs, 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 hybridises general tabu search with some heuristic search strategies. In [17], the nurse rostering problem was formulated into a multi-criteria 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 Dowsload [1] 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 [2] which utilised a different solution encoding/decoding scheme and some specialised 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 [1]. 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], [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 UK hospital, originally studied in [15] and [1]. The formulation employed in those two studies represents a typical 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 (7 day shifts and 7 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 minimise the following cost function

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

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

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

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 optimisation 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 "survive of the fittest". For an optimisation 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 specialised 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 post-reparation, 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 [1], [19], [28]–[30]. The idea is to transform the constrained optimisation problem into an unconstrained one by introducing a penalty term into the objective function to penalise 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):

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

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 optimisation 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 penalise 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 twenty 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 multi-objective optimisation 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 optimisation 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 favours 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 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 [1] 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)

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

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 non-improvement generations or the optimal solution is reached
P_f	0.25
Selection	Tournament selection with stochastic ranking (S=7) + elitism

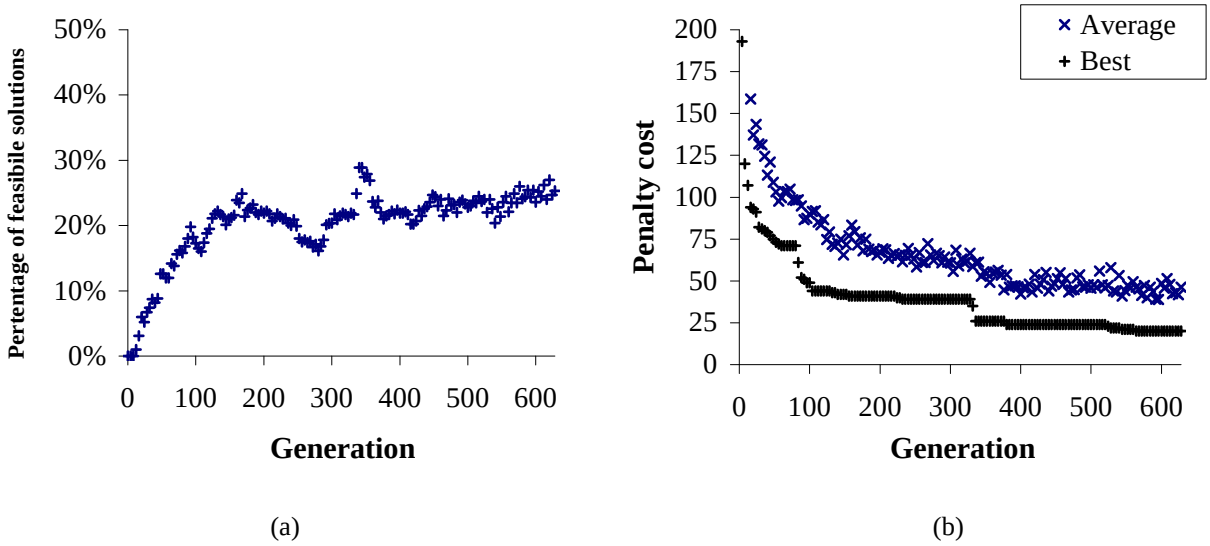

Fig. 1. Analyses of the population evolution of the stochastic ranking genetic algorithm. (a): dynamics of the percentage of feasible solutions in the population. (b): Average and best penalty cost of the population over time.

while in [1] the best 10% of solutions are directly copied to the next generation. In the same way as in [1], 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.

Figs. 1(a) and 1(b) present 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 stabilises at between 20% to 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.

Figs. 2(a) and 2(b) present the results of the stochastic ranking genetic algorithm (SRGA) in comparison with the indirect genetic algorithm (IGA)[3] in [2] among 20 independent runs. Due to space limitation, we do not compare with the results in [1] but they are inferior to those in [2] 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

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

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

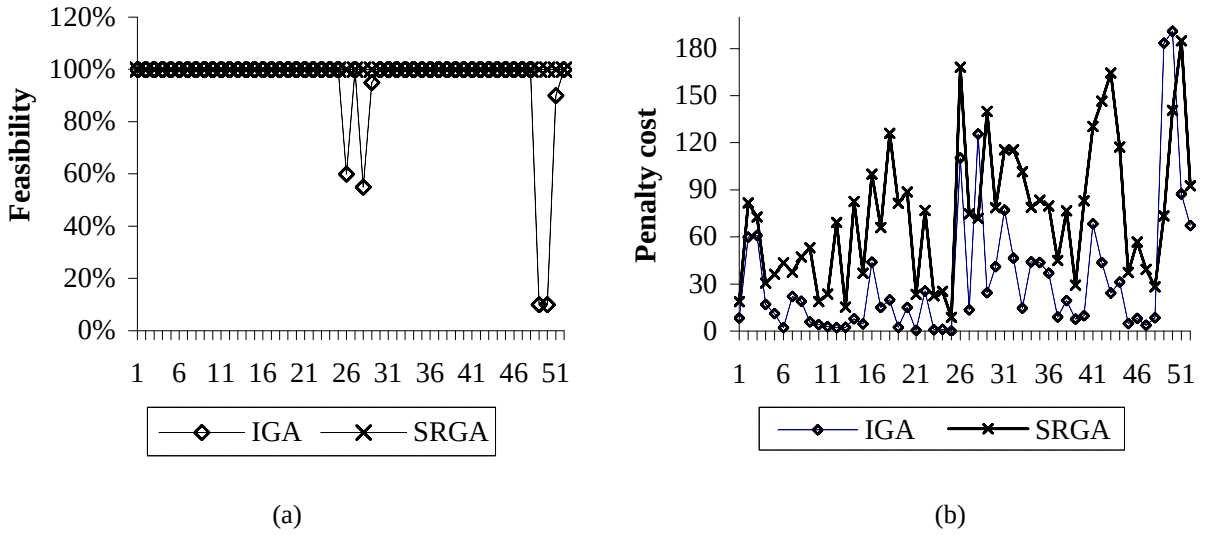

Fig. 2. A comparison of stochastic ranking genetic algorithm (SRGA) with indirect genetic algorithm (IGA) by [2]: (a) Percentages of feasible solutions obtained among 20 runs. (b) Average objective value over 20 runs.

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 hybridise the SRGA with a recently proposed simulated annealing hyper-heuristic algorithm [5].

V. Hybridisation with a Simulated Annealing Hyper-heuristic

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 hybridise the genetic algorithm with a simulated annealing hyper-heuristic approach that has demonstrated impressive performance over three difficult optimisation 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 simulated annealing hyper-heuristic algorithm into the genetic algorithm. Firstly, it is computationally expensive to execute a simulated annealing hyper-heuristic 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 pseudo-code for the proposed algorithm is outlined in Fig. 3.

The parameters of the genetic algorithm remain the same except that the population size is decreased to 100 for computational considerations. The parameters with regard to the simulated annealing hyper-heuristics are set as follows. $K = 20$, $t_s = 10.0$, $t_f = 0.1$, $\beta = (t_s - t_f)/(gen' \cdot t_s \cdot t_f)$, 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 simulated annealing hyper-heuristic where the low level heuristics are selected uniformly (see Fig. 4). We also carried out some experiments on a hybridisation 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 neighbouring solution, the new solution is accepted according to the simulated annealing acceptance criteria. A

step1:	Initialisation: 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. Pseudo-code 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 $LS\_SAHH(I_u, K)$

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 'ist improving f'.
- H_5 Same as H_1 but 'ist 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 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 [1], with each instance being solved 20 times using independent random seeds. The detailed results of the hybrid algorithm are presented in tables III, IV and Fig. 7. We now make comparisons with the tabu search hyper-heuristic (TSHH) [19] [4], the indirect genetic algorithm (IGA) [2], and a more recently proposed estimation of distribution algorithm (EDA) [23]

Table II presents a comparison of the average objective values by the proposed hybrid algorithm and those by TSHH in [38]. It can be seen that for the majority of instances (33 out of 52), the hybrid algorithm performed better than TSHH. TSHH produced better results for 5 instances only and for the remaining 13 instances both algorithms obtained the same results.

The detailed results of IGA and EDA are presented in Fig. 5 and Fig. 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 IP package which is impractical due to the high computational and financial costs [2].

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. For 44 out of 52 instances, the hybrid algorithm obtained a good quality solution (i.e.

[4]Due to the unavailability of the detailed computational results in [19], instead we drew results from [38] by the same algorithm with the best parameter configuration HH1:4L.

TABLE II
THE HYBRID ALGORITHM VS THE TABU SEARCH HYPER-HEURISTIC

Set	1	2	3	4	5	6	7	8	9	10	11	12	13
TSHH	**8.0**	51.3	**50.0**	**17.0**	**11.0**	**2.0**	11.7	16.0	3.3	**2.1**	**2.0**	2.1	**2.0**
Hybrid Algorithm	**8.0**	**49.9**	**50.0**	**17.0**	**11.0**	**2.0**	**11.0**	**14.1**	**3.0**	2.6	**2.0**	**2.0**	**2.0**
Set	14	15	16	17	18	19	20	21	22	23	24	25	26
TSHH	3.3	**3.0**	39.0	13.3	22.6	1.4	9.2	**0.0**	**25.1**	0.3	**1.0**	**0.0**	**48.1**
Hybrid Algorithm	**3.2**	**3.0**	**38.4**	**9.0**	**18.0**	**1.3**	**8.9**	**0.0**	25.3	**0.1**	**1.0**	0.5	73.0
Set	27	28	29	30	31	32	33	34	35	36	37	38	39
TSHH	**3.3**	64.3	**15.0**	35.1	68.3	40.5	11.0	41.7	38.7	34.3	5.2	**13.0**	**5.0**
Hybrid Algorithm	3.7	**63.1**	15.1	**35.0**	**65.2**	**40.0**	**10.8**	**38.0**	**36.0**	**32.1**	**5.0**	**13.0**	**5.0**
Set	40	41	42	43	44	45	46	47	48	49	50	51	52
TSHH	8.6	60.7	48.6	26.4	30.1	**3.0**	6.5	3.4	5.6	29.8	108.9	74.2	61.5
Hybrid Algorithm	**7.2**	**58.0**	**38.3**	**22.0**	**23.2**	**3.0**	**4.0**	**3.0**	**4.1**	**27.0**	**107.3**	**74.0**	**58.0**

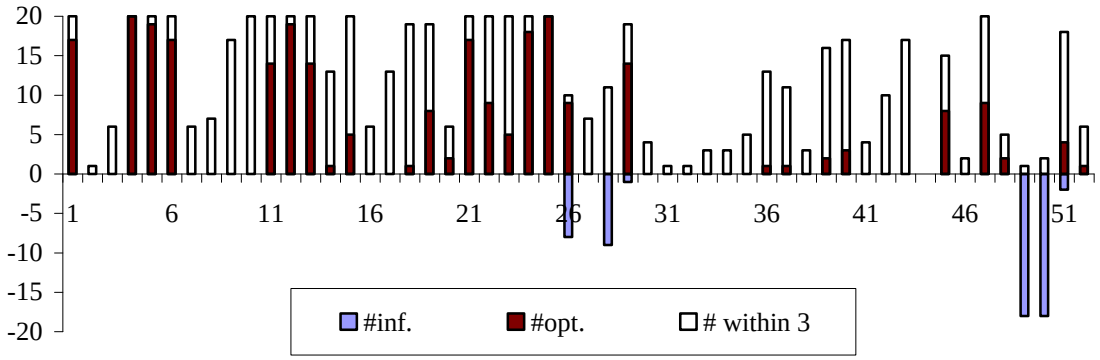

Fig. 5. Detdetailed results by IGA

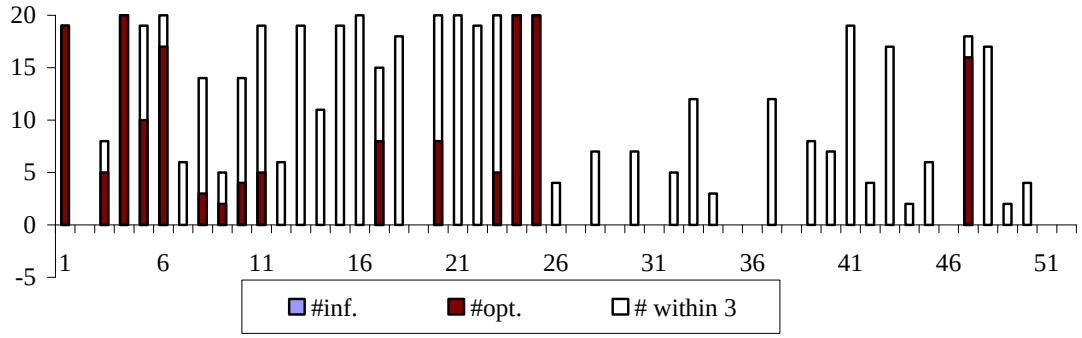

Fig. 6. Detail results by EDA with ant-miner

solutions within 3 penalty cost away from optimality) on each of 20 independent runs.

In terms of computational time, IGA is very fast, with an average time of 9.3 seconds per instance on a Pentium II PC. EDA takes an average of 30 seconds on a Pentium IV 2.0GHz PC with 512MB RAM. The average computational time for our hybrid algorithm is 61.2 seconds on a Pentium IV 1.8 PC with 2GB RAM. This is mainly due to the local search phase SAHH_LS, which is computationally expensive. TSHH was run on a PC Pentium III 1000Mhz with 128MB RAM with CPU time limit of 60 seconds. Nevertheless, with extra computational time, the hybrid algorithm can produce much better results than both the IGA and the EDA heuristic and the TSHH for most of problem instances. Table III presents more detailed results of the proposed hybrid algorithm in comparison with the simulated annealing hyper-heuristic (SAHH) in [5]. The performance of the hybrid algorithm and SAHH is similar for several problem instances. However, the

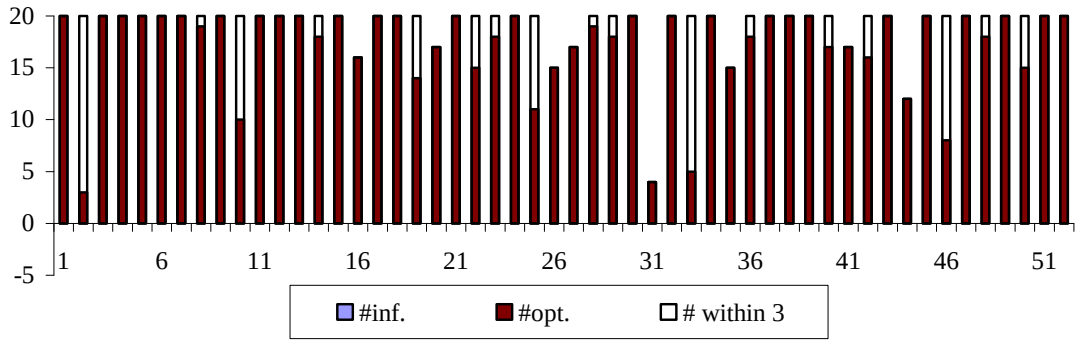

Fig. 7. Detailed results by the hybrid algorithm

TABLE III
HYBRID ALGORITHM VS SAHH (CONTINUED ON NEXT PAGE)

	IP	**SAHH**				**Hybrid Algorithm**			
Set		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	49	49.9	51	0.45
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	2.0	2	0.00
7	11	11	11.0	11	0.00	11	11.0	11	0.00
8	14	14	14.1	15	0.31	14	14.1	15	0.22
9	3	3	3.0	3	0.00	3	3.0	3	0.00
10	2	2	2.5	4	0.69	2	2.6	4	0.68
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	3.2	5	0.49
15	3	3	3.0	3	0.00	3	3.0	3	0.00
16	37	37	41.4	66	8.66	37	38.4	44	2.87
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	1.3	2	0.47
20	7	7	8.3	21	3.18	7	8.9	20	4.64
21	0	0	0.2	1	0.41	0	0.0	0	0.00
22	25	25	25.3	27	0.57	25	25.3	27	0.57
23	0	0	0.2	1	0.41	0	0.1	1	0.31
24	1	1	1.0	1	0.00	1	1.0	1	0.00
25	0	0	0.2	1	0.41	0	0.5	1	0.51

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, 26, 33 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 3.9% (respectively 20.1% and 47.7%) compared with SAHH.

Another advantage of the proposed hybrid approach is its simplicity of implementation and flexibility to be adapted to other constrained optimisation problems. Compared with the penalty function method and other constraint handling techniques, stochastic ranking is simpler and more generic.

TABLE IV
HYBRID ALGORITHM VS SAHH (CONTINUED FROM PREVIOUS PAGE)

	IP	**SAHH**				**Hybrid Algorithm**			
Set		best	mean	worst	stdev	best	mean	worst	stdev
26	48	48	91.1	198	54.04	48	73.0	148	44.43
27	2	2	4.4	13	4.49	2	3.7	13	4.03
28	63	63	63.3	65	0.55	63	63.1	64	0.22
29	15	15	15.3	18	0.72	15	15.1	16	0.31
30	35	35	35.7	40	1.42	35	35.0	35	0.00
31	62	62	64.7	66	1.84	62	65.2	66	1.64
32	40	40	40.1	41	0.22	40	40.0	40	0.00
33	10	10	15.6	103	20.64	10	10.8	11	0.44
34	38	38	38.1	39	0.22	38	38.0	38	0.00
35	35	35	35.9	39	1.23	35	36.0	39	1.78
36	32	32	32.7	33	0.49	32	32.1	33	0.31
37	5	5	5.0	5	0.00	5	5.0	5	0.00
38	13	13	13.1	15	0.45	13	13.0	13	0.00
39	5	5	5.0	5	0.00	5	5.0	5	0.00
40	7	7	7.5	9	0.69	7	7.2	8	0.37
41	54	54	61.7	83	12.11	54	58.0	82	9.78
42	38	38	38.8	40	0.55	38	38.3	40	0.55
43	22	22	23.2	32	3.04	22	22.0	22	0.00
44	19	19	27.3	34	5.21	19	23.2	30	5.22
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	0.89
47	3	3	3.0	3	0.00	3	3.0	3	0.00
48	4	4	4.3	5	0.44	4	4.1	5	0.31
49	27	27	31.9	118	20.28	27	27.0	27	0.00
50	107	107	107.9	109	0.81	107	107.3	108	0.44
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**	**21.1**	**22.1**	**25.0**	**1.58**

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 simulated annealing hyper-heuristic 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 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 a simulated annealing hyper-heuristic. In this algorithm, a stochastic ranking method was used to improve the constraint handling capability of the genetic algorithm while a simulated annealing hyper-heuristic 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 has demonstrated better performance with regard to feasibility. To improve the solution quality in terms of the objective function, a simulated annealing hyper-heuristic algorithm was hybridised 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 and efficient hybrid algorithm for the nurse rostering problem. The algorithm is also simple and flexible and provides significant potential for extension to other constrained optimisation problems.

REFERENCES

[1] U. Aickelin and K. A. Dowsland, "Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem," *Journal of Scheduling*, vol. 3, pp. 139–153, 2000.

[2] U. Aickelin and K. A. Dowsland, "An indirect genetic algorithm for a nurse scheduling problem," *Computers & Operations Research*, vol. 31, no. 5, pp. 761–778, 2003.

[3] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2000.

[4] W. E. Hart, N. Krasnogor, and J. Smith, Eds., *Recent Advances in Memetic Algorithms*, ser. Studies in Fuzziness and Soft Computing. Springer, 2004, vol. 166.

[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, University of Nottingham, UK, Tech. Rep., 2007.

[6] E. K. Burke, P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem, "The state of the art of nurse rostering," *Journal of 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," *Journal of the Society for Health Systems*, vol. 2, no. 1, pp. 62–72, 1990.

[8] R. Hung, "Hospital nurse scheduling," *Journal of Nursing Administration*, vol. 25, pp. 21–23, 1995.

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

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

[11] J. P. Howell, "Cyclical scheduling of nursing personnel," *Hospitals*, vol. 40, pp. 77–85, 1966.

[12] D. W. Warner, "Scheduling nurse personnel according to nursing preference: A mathematical programming approach," *Operations Research*, vol. 24, pp. 842–856, 1976.

[13] H. E. Miller, W. P. Pierskalla, and G. Rath, "Nurse scheduling using mathematical programming," *Operations Research*, 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," *Journal of Operational Research Society*, vol. 51, pp. 393–4, 2000.

[15] K. A. Dowsland, "Nurse scheduling with tabu search and strategic oscillation," *European Journal of Operational Research*, vol. 106, 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 *Simulated Evolution and Learning: Second Asia-Pacific Conference on Simulated Evolution and Learning, SEAL'98*, ser. Lecture Notes in Computer Science. Springer, 1999, vol. 1585, pp. 187–194.

[17] E. K. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe, "A multi criteria meta-heuristic approach to nurse rostering," in *Thrid Practice and Theory of Automated Timetabling International Conference, PATAT2000*, ser. Lecture Notes in Computer Science. Springer, 2001, vol. 2079, 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," *Applied Artificial Intelligence*, 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," *Journal of 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," *European Journal of Operational Research*, vol. 175, pp. 649–671, 2006.

[21] P. D. E. K. Burke, P. I. Cowling and G. Vanden Berghe, "A memetic approach to the nurse rostering problem," *Applied Intelligence*, vol. 15, no. 3, pp. 199–214, 2001.

[22] U. Aickelin and J. Li, "An estimation of distribution algorithm for nurse scheduling," *To apprear in Annals of Operations Research*, 2006.

[23] U. Aickelin, E. K. Burke, and J. Li, "An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering," *To appear in Journal of the Operational Research Society*, 2006.

[24] J. H. Holland, *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press, 1975.

[25] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[26] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of Heuristics*, vol. 2, pp. 5–30, 1996.

[27] P. Chu and J. E. John, "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, vol. 4, 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 Journal on Computing*, vol. 8, no. 2, pp. 173–182, 1996.

[29] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, pp. 311–338, 2000.

[30] S. Venkatraman and G. G. Yen, "A generic framework for constrained optimization using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 4, pp. 424–435, 2005.

[31] R. Farmani and J. A. Wright, "Self-adaptive fitness formulation for constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 5, pp. 445–455, 2003.

[32] P. D. Surry and N. J. Radcliffe, "The comoga method: Constrained optimization by multi-objective genetic algorithms," *Control and Cybernetics*, vol. 26, no. 3, pp. 391–412, 1997.

[33] F. Y. Cheng and D. Li, "Multiobjective optimization design with pareto genetic algorithm," *Journal of Structural Engineering*, vol. 123, no. 9, pp. 1252–1261, 1997.

[34] C. A. C. Coello, "Treating constraints as objectives for single-objective evolutionary optimization," *Engineering Optimization*, 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 Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 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. 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. Springer, 2005, ch. 17, pp. 529–556.

[38] E. Burke and E. Soubeiga, "Scheduling nurses using a tabu search hyperheuristic," in *The Proceedings of 1st Multidisciplinary International Conference on Scheduling: Theory and Applications*, vol. 1, 2003, pp. 197–218.