Memetic Algorithms for Nurse Rostering

Ender Özcan

Yeditepe University,
Department of Computer Engineering,
Kayisdagi, Istanbul, Turkey
eozcan@cse.yeditepe.edu.tr

Abstract. Nurse rostering problems represent a subclass of scheduling problems that are hard to solve. The goal is finding high quality shift and resource assignments, satisfying the needs and requirements of employees as well as the employers in healthcare institutions. In this paper, a real case of a nurse rostering problem is introduced. Memetic Algorithms utilizing different type of promising genetic operators and a self adaptive violation directed hierarchical hill climbing method are presented based on a previously proposed framework.

1 Introduction

Timetabling problems are well known NP complete problems [15]. As a timetabling problem, shift scheduling is concerned with the arrangement of employee timetables, considering the constraints provided by employees, employers and even customers. A nurse roster is a timetable consisting of shift assignments and rest days of nurses working at a hospital. In nurse rostering, the ultimate aim is to create high quality timetables, taking well-being of nurses as a basis without discarding the concerns of employers.

There is variety of approaches used for solving nurse rostering problems ([10], [12], [23]). Increasing number of researchers applies Genetic Algorithms (GAs) or other metaheuristic approaches, such as, Simulated Annealing, Tabu Search and their hybrids to tackle timetabling problems ([5], [6], [14], [17]). Ahmad et. al. [1] applied a modified version of a GA, named as population-less cooperative genetic algorithm on a 3-shift problem. Kawanaka et. al. [21] used GA to obtain optimal nurse schedules satisfying absolute and desirable constraints. Aickelin et. al. [2] utilized a coevolutionary pyramidal GA for solving nurse rostering. Each subpopulation attempts to solve nurse rostering for a set of nurses having either the same grade or a predetermined combination of them, organized in a hierarchical way as a pyramid for mate selection. Aickelin et. al. [3] proposed an indirect representation in GA for NRP and three different decoders. Recently, research on timetabling started to move towards finding a good hyper-heuristic ([9], [11], [20]); a heuristic for selecting a heuristic among a set of them to solve an optimization problem.

Details about nurse rostering, such as, constraint categorizations, models and approaches can be found in [7], [16] and [27]. In this paper, a set of memetic algorithms (MAs), combining GAs utilizing a set of genetic operators and a self adaptive violation directed hierarchical hill climbing method (VDHC) are introduced. MAs are based on the very same framework proposed by Alkan et. al. [4]. Extensive experiments are performed using randomly generated data and a real one retrieved from a major hospital. VDHC is a promising approach.

2 Nurse Rostering Problem

Nurse rostering problems (NRPs) are constraint optimization problems that can be represented by a 3-tuple $<V, D, C>$. V is a finite set of variables, possibly each representing a shift of a nurse at a hospital, $V = \{v_1, v_2, \ldots, v_i, \ldots, v_N\}$, $D=\{ d_1, d_2, \ldots, d_i, \ldots, d_N\}$, is a finite set of domains of variables, where d_i is the domain of the variable v_i. Let $T=\{ t_1, \ldots, t_j, \ldots, t_M\}$ represent a set of start times for a shift, then a possible domain of each variable: $d_i \subseteq T$. C is a set of constraints to be satisfied, $C=\{ c_1, c_2, \ldots, c_L\}$. NRP can be described as a search for finding the best assignment (v_i, t_j) for each variable $v_i \in V$, such that, all constraints are satisfied. The assignment implies that the i^{th} shift of a nurse at v_i starts at t_j. Constraints are categorized as *hard* or *soft*, where hard constraints must be satisfied and soft ones represent preferences.

2.1 Nurse Rostering Problem at Memorial Hospital

Shift schedules in Memorial Hospital (Istanbul, Turkey) are generated manually for all the departments in the hospital. There are two shift periods per day: *day* and *night*. In order to simplify the timetabling process, the hospital authorities produce a weekly schedule manually, although a biweekly schedule is preferred. Since the preferences of nurses are essential and might change in time, schedules are acyclic. There are three departments and about twenty nurses in the hospital. In some cases, a nurse from a different department is allowed to work at another department for support. Nevertheless, this type of cross duty does not occur often. Each nurse is considered to be independent belonging to a department. A nurse has a rank assigned from {0, 1, 2} indicating the level of experience (from lowest to highest). Rank 2 implies an experienced nurse. There are one or two nurses with rank 2 at each department. During the analysis a set of hard and soft constraints are determined. Hard Constraints:

- *Presets* (PRC): Presets represent the predetermined shift schedules of nurses.
- *Shift Constraint* (SHC): At a department, during each shift there must be at least one nurse.
- *Successive Night Shifts Constraint* (SNC): A nurse can not be assigned to more than two successive night shifts.
- *Successive Day Shifts Constraint* (SDC): A nurse can not be assigned to more than three successive day shifts.

– *Successive Shifts Constraint* (SSC): A nurse can not be assigned to two successive shifts. A day shift in one day and a night shift in the following day are considered as successive shifts.
– *Exclude Night Shifts Constraint* (ENC): Night shifts can not be assigned to an experienced nurse with rank 2.
– *On-duty Constraint* (ODC): Each nurse can not be assigned less than eight shifts per two weeks.

Soft Constraints:

– *Off-duty Constraint* (RDC): Nurses can define at most 3 rest day preferences.

3 Memetic Algorithms for Solving Nurse Rostering Problems

Genetic Algorithms (GAs) were introduced by J. Holland [19], and have been used to solve many difficult problems [18]. Usefulness of hill climbing in population based algorithms is emphasized by many researchers ([24, 28, 29]).

The problem described in Section 2 is chosen, due to the similarities with the university course scheduling problem, which is described in [25]. Memetic Algorithms (MAs) are presented for solving nurse rostering problems, based on a violation directed hierarchical hill climbing (VDHC). In most of the timetabling problem instances, variables are arranged hierarchically. Let a *classifier* be a subset of variables, then at each level in the hierarchy; a set of classifiers, representing logical groupings can be formed. In most of the cases, classifiers at a hierarchy level are collectively exhaustive in *V*. Classifiers form a basis for designing of a rich set of operators, discussed in the following sections. Arrangements can be formed statically or dynamically. A *static* arrangement is used during the experiments (Fig. 1). This study is a part of an attempt to provide a framework for solving different type of timetabling problems using a single tool based on Memetic Algorithms. Ozcan proposed an XML standard for timetabling problems in [27]. The goal is to represent different classes of timetabling problems using a single format. There are some developers already supporting the XML standard; schoolTool (http://www.schooltool.org), tablix (http://www.tablix.org).

3.1 Representation

The direct representation is used. Assuming *S* denotes the total number of nurses in a hospital, there is a subset of nurses available for duty at each department. Each nurse has a timetable, having *R* slots (days) to be filled with a shift type. *R* is set to 14 for producing bi-weekly acyclic schedule for each nurse. Each gene denotes the start time of a shift for a nurse in a day. Additional to day shift (**1**) and night shift (**2**), off-duty (**0**) allele is used in the individual representation. The representation scheme allows implementation of different sets of genetic operators.

Shifts of all nurses (variables) are arranged hierarchically in the individual representation (Fig. 1). Top hierarchy level, denoted as H-level contains a single classifier; *V*. There are three more hierarchy levels. D-level, N-level and G-level contain *P*, *S* and

N=SxR number of classifiers, respectively, as demonstrated in Fig. 1. Furthermore, each classifier at a level is a partition of V.

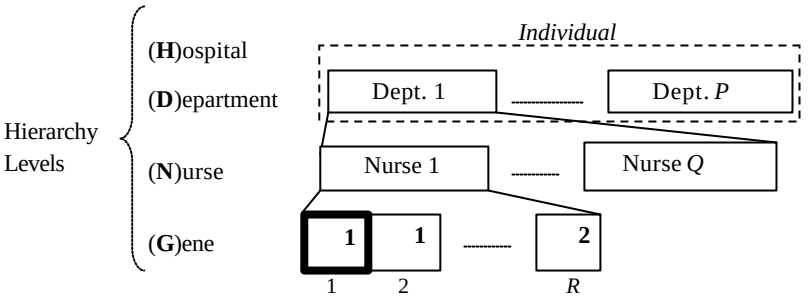

Fig. 1. Individual representation used in MA for solving NRP

3.2 Fitness Function

An optimum nurse roster is the one with no violations. Let NR represent a proposed schedule of all nurses in the hospital, $T_i \in NR$ represent the nurse roster of the i^{th} nurse, $p_j(.)$ represent the violation penalties due to the j^{th} constraint for a given nurse roster and w_j to be the associated weight for the corresponding type of constraint. Fitness accumulates each weighted violation penalty with respect to its constraint type.

$$f(NR) = \sum_{\forall i \forall j} w_j p_j(T_i) \quad \textbf{(1)}$$

3.3 Mutation

Mutation can be applied on all classifiers at the same level, independently. For example, at D-level, the same mutation operator can be applied within all the departments using an appropriate mutation rate, as if each department is an individual. Similarly, at N-level, the mutation operator can be applied on each nurse. Swap mutation can be extended and a part (or whole) of the hierarchy level component can be replaced by a same size component at the same level. This set of mutations is straightforward to implement, if hierarchy level components are partitions. Behave as if each partition is a gene and swap it. For example, two nurse rosters or 3 days of the two nurse rosters can be swapped using an appropriate mutation rate. Similarly, two department rosters, or the same number of several nurse rosters in two departments can be swapped. Representation allows designing violation directed mutations as well:

- At a hierarchy level, select a classifier (partition or subset) based on the violations
- Apply mutation operator only on the selected classifier

As a different approach, the same or an adaptively selected mutation operator can be applied on each classifier, separately. Traditional mutation operator (M0) is used in

the experiments. M0 randomly perturbs an allele with a *mutation rate* of 1/*length_of_the_part_to_be_mutated*. Furthermore, two *smart* mutation operators are implemented. M1 and M2 operate in a similar manner on nurses and departments, respectively. Two randomly selected classifiers enter into a tournament. The one causing higher number of violations is selected for mutation. M0 is applied on the selected classifier (part of the chromosome). More disruptive mutations are implemented as well. M3 and M4 apply M0 on each nurse and department, respectively.

3.4 Crossover

Traditional one point crossover (1PTX) and uniform crossover (UX) are used to design a set of modified crossover operators. Boundary based crossover operators are applied on the classifiers defined at a hierarchy level. For example, 1PTX_N works at the N-level and 1PTX is applied as if the set of genes forming a nurse classifier is itself a gene. Hence, 6 different types of crossover operators are implemented: 1PTX (X1), UX (X2), 1PTX_D (X3), UX_D (X4), 1PTX_N (X7), and UX_N (X8).

Less disruptive and *smart* crossovers can be created to operate in two steps:

- Select one of the classifiers based on a strategy
- Apply crossover only within that classifier

As a result, four more crossover operators are implemented, utilizing tournament selection strategy with a tour size of two. A classifier is selected by comparing the total number of violations in each classifier at a level. 1PTX or UX is used on the selected classifier: 1PTX_SD (X5), UX_SD (X6), 1PTX_SN (X9), UX_SN (X10).

Additionally, highly disruptive crossover operators are created, applying crossover on all classifiers at a level, one by one. 1PTX_AD (X11) applies 1PTX on all departments, while 1PTX_AN (X12) applies 1PTX on all nurses. More crossover techniques can be generated allowing different crossovers to be operational at each classifier in the same level. For example, while 1PTX can be applied on the first and fourth nurse schedules, UX can be applied on the second and third nurse schedules, assuming four nurses and a single department. Ultimately, this type of strategies might require an adaptive method to decide which crossover to apply.

3.6 Hill Climbing

Timetabling problems are also formulated as multi-criteria optimization problems. This formulation would be very useful, especially in the existence of different types of soft constraints. Several solutions might be obtained having comparable qualities. Obviously, while attempting to reduce the violations due to a constraint, overall quality of a suggested solution might worsen. Applying a hill climbing method as a part of a hybrid algorithm is computationally expensive in timetabling problems. After each step is applied, the new configuration has to be evaluated in order to determine whether an improvement is provided or not. Yet, if the quality of solution increases, a hill climbing approach might be preferred. A violation directed hierarchical hill climbing method, denoted as VDHC is proposed as a part of a Memetic Algorithm for solving time-

tabling problems. Hill climbing is applied after mutation. VDHC provides cooperation of a set of hill climbers.

4 Self Adaptive Violation Directed Hierarchical Hill Climbing

VDHC represents a self adaptive approach which requires iterative application of a hill climbing method for a selected type of constraint as shown in Fig. 2. First, hierarchy levels to be used in VDHC are determined with an uppermost level chosen as a starting level to operate on. VDHC stays at a level as long as current candidate solution improves. It applies a selected hill climbing method, evaluating violations due to each constraint type, to a selected classifier at a level. If no improvement is confirmed, then VDHC reduces the area of concern to classifiers at one level down in the hierarchy. Hence, hierarchy level changes and the same steps of the algorithm repeat. VDHC terminates whenever a maximum number of steps is exceeded. Violation based selection methods are suggested.

```
1. Mark the hierarchy levels to be used
2. Set current level to the top hierarchy level
3. while (terminationCriteria-1 are not satisfied) do
   a. while (terminationCriteria-2 are not satisfied)
      do
        i. Start the traversal from the top until to
           the current level and select a classifier
       ii. Select a constraint type
      iii. Apply hill climbing for the selected con-
           straint type within the selected classifier
   b. end while
   c. Lower the hierarchy level
```

Fig. 2. Pseudo-code of the VDHC approach

4.1 VDHC for Nurse Rostering

Three hierarchy levels are marked: H-level, D-level and N-level. As a classifier and constraint selection method a tournament selection is used. At the H-level classifier selection method always returns the whole chromosome. At the D-level, classifier selection method computes the violation contribution of each department to the overall fitness and selects one of them using tournament. Similarly, at the N-level, classifier selection method computes the violation contribution of each nurse to the overall fitness and selects one of them. In order to select a classifier from the N-level, a department has to be determined. Hence, D-level classifier selection is done first. Then the violations caused by each constraint type are distinguished. One of the constraint

types is selected, giving a higher chance to the hill climbing step of the related constraint type causing more violations. Selected hill climbing is applied to the predetermined classifier to get rid of all the violations due to the related constraint type, producing a new individual.

Seven constraint based hill climbing methods are developed corresponding to each constraint type; SHC_HC, SNC_HC, SDC_HC, SSC_HC, ENC_HC, ODC_HC, RDC_HC. PRCs are handled by fixing assignments of related nurse shifts; hence, this constraint does not require application of a hill climbing method. SHC_HC checks departmental rosters and locates shifts without a nurse assignment. Then a nurse in the department is selected randomly and assigned to that shift. SNC_HC checks whether three consecutive shifts are night shifts or not. If they are, one of the shifts is changed to a day shift or marked as off-duty. SDC_HC checks whether four consecutive shifts are day shifts or not. If they are, one of the shifts is modified to a night shift or marked as off-duty. SSCs are partially satisfied by the use of the representation. If a candidate solution contains a successive two day pattern night shift-day shift, SSC_HC modifies the second day shift as either a night shift or off-duty. ENC_HC transforms a night shift assigned to a nurse with rank 2 to a day shift or off-duty. ODC_HC modifies required number of off-duty assignments to either a day or a night shift. RDC_HC attempts to realize nurse preferences. All choices and modifications in each hill climbing method are carried out randomly. In the tests the maximum number of hill climbing steps is a factor of chromosome length.

5 Experiments

A random nurse rostering problem instance generator (RNR) is implemented. 9 problem instances, produced by RNR, and a real data obtained from Memorial Hospital, labeled as *rnd#id* and *mhtr*, respectively, are used in the experiments. Characteristics of the data set are summarized in Table 1. All runs are repeated 50 times. Pentium IV 2 GHz. machines with 256 MB RAM are used. Experiments are performed in three stages. In the first stage crossover operators are compared using *rnd*1-6 data. Operators are compared based on their ranks considering the number of violations of best achieved solutions averaged over runs. In the second stage, mutation operators are tested with the top crossover operator on the same data. In the final stage, all the data set is tested using the best MA. Experimental data can be reached at http://cse.yeditepe.edu.tr/~eozcan/TTML.

Population is initialized randomly, and its size is a factor of the chromosome length. As a mate selection method linear ranking strategy is preferred, giving four times higher chance for the best individual than the worst one to be selected. All the runs are terminated whenever a fitness value of 0 is achieved, or whenever a maximum number of generations is exceeded. It is known that an optimal schedule is possible for the data used in the experiments. Hence, soft and hard constraints are not distinguished. Weight of each penalty is set to 1. Define s*uccess rate* (*s.r.*) indicate the proportion of the successful runs yielding optimal solutions. As a replacement strategy, trans-

generational MA (TGMA) with weak elitism is preferred, based on our previous experience ([4], [25], [26], [28]). Two best individuals are inherited to the next generation and the rest of them are obtained from the offspring pool.

Table 1. Characteristics of the data set used in the experiments. Number of departments and nurses are denoted as *ndep* and *nnur*, respectively. Percentage of nurses from each rank and average number of off-duty preferences of each nurse is denoted as *pnr* and *avrpr*, respectively

label	***mhtr***	***rnd*1**	***rnd*2**	***rnd*3**	***rnd*4**	***rnd*5**	***rnd*6**	***rnd*7**	***rnd*8**	***rnd*9**
ndep	4	3	3	3	4	4	4	6	8	6
nnur	20	21	21	21	21	21	21	34	51	66
***pnr*0**	0.33	0.42	0.18	0.28	0.14	0.19	0.13	0.18	0.19	0.36
***pnr*1**	0.48	0.32	0.51	0.42	0.47	0.46	0.47	0.47	0.47	0.35
***pnr*2**	0.19	0.28	0.32	0.32	0.42	0.37	0.42	0.38	0.35	0.30
avrpr	0.55	1.95	0.67	2.19	1.67	2.33	0.95	1.97	1.88	2.09

In the first stage of experiments M0 is fixed as a mutation operator. According to the results, 1PTX performs better than the rest of the crossover operators (Table 2). UX is the second best. All crossover operators, other than 1PTX and UX fail to find the optimal solution. Considering boundary based and smart crossover methods, the ones operating on N-level perform better. X3 and X4 are the worst crossover methods. X10, X6, X11, X12 and X9 are the top five crossovers in the given order following the traditional operators. Whenever these top crossover operators are used, on average less than 21 violations are left unresolved.

Table 2. Results of the first stage experiments, indicating the rank of each crossover operator

Label	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X8**	**X9**	**X10**	**X11**	**X12**
*rnd*1	1	2	11	12	9	4	10	8	7	3	6	5
*rnd*2	1	1	10	11	8	3	9	7	5	2	4	6
*rnd*3	1	1	10	11	8	3	9	7	6	2	5	4
*rnd*4	1	2	11	12	9	4	10	8	7	3	5	6
*rnd*5	1	2	11	12	9	4	10	6	8	3	5	7
*rnd*6	1	2	11	12	9	4	10	6	7	3	5	8

In the second stage experiments 1PTX is fixed as a crossover operator. According to the experimental results, performance of operators are from the best towards the worst is M0, M1, M2, M4 and M3 (Table 3). M0 and M1 perform approximately the same, while M3 is the worst mutation operator, failing to find the optimal solution in most of the cases. In the last generations, a single violation is left to be resolved for all mutations, except M3. Violation directed smart mutation operators turn out to be more effective than the crossover operators.

TGMA performs best whenever M0 and X1 are used. TGMA reduces the number of violations rapidly in few hundreds of generations on average as established in Table 4.

A run of the best MA takes less than 3 minutes on average for all data. Real data turns out to be the hardest problem instance of all. Genetic Algorithm version of the best MA without VDHC is applied to the real data. MA outperforms the GA version.

Table 3. Results of the 2nd stage experiments, indicating *s.r.* of each mutation operator

Label	M0	M1	M2	M3	M4
*rnd*1	**0.99**	0.96	0.56	0.00	0.48
*rnd*2	**1.00**	**1.00**	**1.00**	0.08	0.90
*rnd*3	**1.00**	**1.00**	**1.00**	0.00	0.84
*rnd*4	**0.98**	**0.98**	0.66	0.00	0.52
*rnd*5	**1.00**	**1.00**	0.94	0.00	0.72
*rnd*6	**1.00**	**1.00**	0.98	0.00	0.86
avr	**0.99**	0.96	0.56	0.00	0.48

Table 4. Results obtained using MA with the best set of operators on the data set

Label	*s.r.*	Avr.Gen./Run	*std.*	Avr.Eval./Gen.	*std.*
mhtr	0.72	1,736	2,881	1,176	1,178
*rnd*1	0.99	219	660	1,293	1,293
*rnd*2	1.00	42	44	1,333	1,333
*rnd*3	1.00	51	53	1,302	1,302
*rnd*4	0.98	265	787	1,306	1,306
*rnd*5	1.00	148	301	1,299	1,299
*rnd*6	1.00	61	62	1,330	1,330
*rnd*7	0.99	181	857	2,092	2,092
*rnd*8	1.00	96	99	3,155	3,155
*rnd*9	1.00	145	167	3,928	3,928

6 Conclusions

Timetabling is an interdisciplinary research area, containing many subclasses, such as, nurse rostering, course timetabling, examination timetabling. As an attempt to propose a general solver for timetabling problems ([4], [25], [26], [27]) a nurse rostering problem is investigated. A real world data obtained from Memorial Hospital and randomly generated data set are used as a test bed. Various mutation and crossover operators, including boundary oriented and smart genetic operators are presented to be used in timetabling problems. Several of these operators and proposed self adaptive violation directed hierarchical hill climbing operator (VDHC) are experimented within Memetic Algorithms. These operators can be used in other approaches as well.

VDHC and suggested operators exploit the underlying structure of problem instances. VDHC boosts the performance of the GA for nurse rostering as expected. Using a hierarchy of levels provides means to correct conflicts once and for all, or for a

group of events, or for a single event. Violation directed operators; especially smart mutations achieve promising performances.

Proposed framework enables researchers to design a variety of operators. Such operators are already used by some researchers. For example, the shake operators suggested in [10] are a subset of genetic operators described for the MAs in Section 3. Other than static arrangement of data, *dynamic* arrangement is also possible. For example, considering a nurse rostering problem, shift assignments of nurses in the same period forms a dynamic arrangement. List of nurses might change from one candidate solution to another. More operators can be designed to work on these dynamic arrangements in a similar manner as discussed for static arrangements. Combining these operators underneath a hyper-heuristic might yield good solutions. As a future work, different combinations of hill climbing methods and genetic operators will be investigated. MA with VDHC will be compared to a multimeme strategy [22].

Acknowledgement

Author thanks Özgür Kelemci for modifying GAlib and obtaining the real data.

References

1. Ahmad, J., Yamamoto, M., and Ohuchi, A.: Evolutionary Algorithms for Nurse Scheduling Problem. Proc. of IEEE Congress on Evolutionary Computation (2000) 196-203.
2. Aickelin, U., and Bull, L.: On the Application of Hierarchical Coevolutionary Genetic Algorithms: Recombination and Evaluation Partners. JASS, 4(2) (2003) 2-17
3. Aickelin, U., and Dowsland, K.: An Indirect Genetic Algorithm for a Nurse Scheduling Problem. Computers & Operations Research, 31(5) (2003) 761-778
4. Alkan, A., and Ozcan, E.: Memetic Algorithms for Timetabling. Proc. of IEEE Congress on Evolutionary Computation (2003) 1796-1802
5. Berrada, I., Ferland, J., and Michelon, P.: A Multi-Objective Approach to Nurse Scheduling eith both Hard and Soft Constraints. Socio-Economic Planning Science. vl. 30(1996)183-193
6. Burke, E.K., De Causmaecker, P., and Vanden Berghe, G.: A Hybrid Tabu Search Algorithm For the Nurse Rostering Problem, Proc. of the Second Asia-Pasific Conference on Simulated Evolution and Learning, vol. 1, Applications IV (1998) 187-194
7. Burke, E.K., De Causmaecker, P., and Vanden Berghe, G., Van Landeghem, H.: The State of the Art of Nurse Rostering, Journal of Scheduling, 7 (2004) 441-499
8. Burke, E.K., Cowling, P.I., De Causmaecker, P., and Vanden Berghe, G.: A Memetic Approach to the Nurse Rostering Problem, Applied Intelligence, vol 15 (2001) 199-214
9. Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S.: Handbook of metaheuristics, chapter 16, Hyper-heuristics: an emerging direction in modern search technology, Kluwer Academic Publisher (2003) 457-474
10. Burke, E.K., De Causmaecker, P., Petrovic, S., Vanden Berghe G.: Variable Neighbourhood Search for Nurse Rostering Problems, in Metaheuristics: Computer Decision-Making (edited by M.G.C. Resende and J. P. de Sousa), Chapter 7, Kluwer (2003) 153-172
11. Burke, E., and Soubeiga, E.: Scheduling Nurses Using a Tabu-Search Hyperheuristic, Proc. of the 1st MISTA, vol. 1 (2003) 197-218

12. Chun, A.H.W., Chan, S.H.C., Lam, G.P.S., Tsang, F.M.F., Wong, J., and Yeung, D.W.M.: Nurse Rostering at the Hospital Authority of Hong Kong, Proc. of 17th National Conference on AAAI and 12th Conference on IAAI (2000) 951-956
13. Downsland, K.: Nurse Scheduling with Tabu Search and Strategic Oscillation, European Journal of Operations Research. Vol. 106, 1198 (1998) 393-407
14. Duenas, A., Mort, N., Reeves, C., and Petrovic, D.: Handling Preferences Using Genetic Algorithms for the Nurse Scheduling Problem, Proc.of the 1st MISTA, vol.1(2003)180-195
15. Even, S., Itai, A., and Shamir, A.: On the Complexity of Timetable and Multicommodity Flow Problems, SIAM J. Comput., 5(4) (1976) 691-703
16. Fang, H.L. Genetic Algorithms in Timetabling and Scheduling, PhD thesis, Department of Artificial Intelligence, University of Edinburgh, Scotland (1994)
17. Gendrau, M., Buzon, I., Lapierre, S., Sadr, J., and Soriano, P. A Tabu Search Heuristic to Generate Shift Schedules, Proc. of the 1st MISTA, vol.2 (2003) 526-528
18. Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading (MA) (1989)
19. Holland, J. H. Adaptation in Natural and Artificial Systems, Univ. Mich. Press (1975)
20. Han, L., and Kendall, G. Application of Genetic Algorithm Based Hyper-heuristic to Personnel Scheduling Problems, Proc. of the 1st MISTA, vol.2 (2003) 528-537
21. Kawanaka, H., Yamamoto, K., Yoshikawa, T., Shinogi, T., and Tsuruoka, S. Genetic Algorithms with the Constraints for Nurse Scheduling Problem, Proc. of IEEE Congress on Evolutionary Computation (CEC), Seoul (2001) 1123-1130
22. Krasnogor, N. Studies on the Theory and Design Space of Memetic Algorithms, PhD Thesis, University of the West of England, Bristol, United Kingdom (2002)
23. Li, H., Lim, A., and Rodrigues, B.: A Hybrid AI Approach for Nurse Rostering Problem, Proc. of the 2003 ACM Symposium on Applied Computing (2003) 730-735
24. Moscato, P., and Norman, M. G.: A Memetic Approach for the Traveling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems, Parallel Computing and Transputer Applications (1992) 177-186
25. Ozcan, E., and Alkan, A.: Solving Time Tabling Problem using Genetic Algorithms, Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (2002) 104-107
26. Ozcan, E., Ersoy, E.: Final Exam Scheduler - FES, 2005 IEEE CEC, (2005) to appear
27. Ozcan, E.: Towards an XML based standard for Timetabling Problems: TTML, Multidisciplinary Scheduling: Theory and Applications, Springer Verlag (2005) 163 (24)
28. Ozcan, E., and Onbasioglu E.: Genetic Algorithms for Parallel Code Optimization, Proc. of 2004 IEEE Congress on Evolutionary Computation, vol. 2 (2004) 1775-1781
29. Radcliffe, N. J., and Surry, P.D.: Formal memetic algorithms, Evolutionary Computing: AISB Workshop, LNCS, vol. 865, Springer Verlag (1994) 1-16
30. Ross, P., Corne, D., and Fang, H-L.: Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation, Proc. of PPSN III (1994) 556-565
31. Ross, P., Corne, D., and Fang, H-L.: Fast Practical Evolutionary Timetabling, Proc. of AISB Workshop on Evolutionary Computation (1994) 250-263
32. De Werra, D.: An introduction to timetabling, European Journal of Operations Research, 19:151-162 (1985)