APPLYING COMPUTATIONAL INTELLIGENCE APPROACHES TO THE STAFF SCHEDULING PROBLEM

Vasileios Perlis, Charilaos Akasiadis and Konstantinos Theofilatos
Department of Computer Engineering and Informatics, University of Patras, 26504 Patra, Greece

Grigorios N. Beligiannis
Dept. of Business Administration of Food and Agricultural Enterprises, Univ. of Western Greece, 30100 Agrinio, Greece

Spiridon D. Likothanasis
Department of Computer Engineering and Informatics, University of Patras, 26504 Patra, Greece

Keywords: Staff scheduling, Heuristics, Genetic algorithm, Particle swarm optimization.

Abstract: Staff scheduling for public organizations and institutions is an NP-hard problem and many heuristic optimization approaches have already been developed to solve it. In the present paper, we present two meta-heuristic computational intelligence approaches (Genetic Algorithms and Particle Swarm Optimization) for solving the Staff scheduling problem. A general model for the problem is introduced and it can be used to express most of real-life preferences and employee requirements or work regulations and cases that do not include overlapping shifts. The Genetic Algorithm (GA) is parameterized, giving the user the opportunity to apply many different kinds of genetic operators and adjust their probabilities. Classical Particle Swarm Optimization (PSO) is modified in order to be applicable in such problems, a mutation operator has been added and the produced PSO variation is named dPSOmo (discrete Particle Swarm Optimization with mutation operator). Both methods are tested in three different cases, giving acceptable results, with the dPSOmo outperforming significantly the GA approach. The PSO variation results are very promising, encouraging further research efforts.

1 INTRODUCTION

Staff scheduling for public organizations and institutions is a hard to solve problem that belongs to the NP-hard class. The big number of employees, the alternating planning period length, the different contract regulations and the personal preferences, are all combined into a big set of constraints, that the schedule designer is challenged to satisfy (Burke et al., 2008).

The quality of the roster or schedule has a large impact on the performance of the institutions and the quality of life of the employees (Baker et al., 2003; Burke et al., 2008). So, efficient planning of it is necessary, in times when requirements are increasing and the cost needs to be kept low.

By using computers and staff scheduling software, the construction of the schedule becomes faster and more efficient, giving the production process a big boost. The use of computers for this matter started from 1970, for smaller problems, but more recent approaches include flexible schedules too. (Cheang et al., 2003).

The problem of staff scheduling has been approached in several ways, using exact or heuristic methods. Any approach could be considered as a decision support system.

Initial approaches often included optimization and mathematical programming. There are examples of linear mathematical programming (Jaumard et al., 1998), integer programming (Bartholdi et al., 1980), goal programming (Ozkarahan and Bailey, 1988) and network programming (Millar and Kiragu, 1998). Such approaches though, are based on the optimization of a single objective function, which makes them inappropriate when the number of constraints is large. We overcome this with the use of multi criteria objectivity and near-optimal methods.

In cases of high complexity, it is necessary to adopt the decision making or constraint satisfaction

DOI: 10.5220/0003669701680173
In *Proceedings of the International Conference on Evolutionary Computation Theory and Applications* (ECTA-2011), pages 168-173
ISBN: 978-989-8425-83-6

perspective. These categories are further divided in systemic search in trees, constraint satisfaction methods, heuristic and meta-heuristic methods and artificial intelligence approaches. Examples of heuristic and meta-heuristic methods are memetic algorithms (Özcan, 2005), genetic algorithms (Easton and Mansour, 1993), simulated annealing (Brusco, 1995) and tabu search (Dowsland, 1998). Artificial intelligence methods include constraint programming (Weil et al., 1995) and expert systems (Chen and Yeung, 1992). (Cheang et al., 2003), (Burke et al., 2004).

Lately, competitions are being organized, like the Nurse Rostering Competition (http://www.kuleuven-kortrijk.be/nrpcompetition), where researchers can present, test and compare their approaches on the same instances of the problem.

In the present paper, we define a model for the staff scheduling problem and we propose two different computational intelligence approaches for solving it, a genetic algorithm approach and a novel particle swarm optimization approach (dPSOmo) with mutation operator whose structure and operator are different from the classical PSO. The model is designed so as to be applicable to more than one cases. We tested the algorithms on published instances of nurse scheduling, coming to quality solutions, especially when using the dPSOmo method.

In the following chapters we present the model for the staff scheduling problem, the two proposed methods, the results of the experiments and the conclusions that accrue.

2 PROBLEM DEFINITION

The staff scheduling problem in a general form can be defined as follows:

Given: A set of employees, a set of shifts, a set of constraints and a planning period, we search for the optimal way to assign shifts over the planning period, in order to satisfy as many constraints as possible. To be more specific, we define some specific notions.

Let the set of n employees be $E = \{e_1, e_2, ..., e_n\}$, the set of p days $D = \{d_1, d_2, ..., d_p\}$ and the set of k shifts $S = \{0, s_1, s_2, ..., s_k\}$, where 0 represents the day-off shift.

We define the shift assignment $u_{e,d}$, $u_{e,d} = s$ if the employee e, at day d is assigned to shift s.

We also define the assignment check function $f(e,d,s)$ which enables us to check if the employee e, the day d, has been assigned to shift s where $e \in E, d \in D, s \in S \cup \{A\}$, where A represents any shift (day-on). So: $f(e,d,s) = \begin{cases} 1, & \text{if } s \neq A \text{ and } u_{e,d} = s \\ 1, & \text{if } s = A \text{ end } u_{e,d} \neq 0 \\ 0, & \text{otherwise} \end{cases}$

Many different types of constraints are analyzed and categorized. Some of them are derived from common sense, such as that an employee cannot work two shifts on the same day, while other are more complex and differentiated for each instance of the problem. The constraints are divided in two categories, the hard and the soft constraints.

Hard constraints must be met for a feasible solution. Each employee must work no more than one shift s per day. That could be expressed: Given an employee e and a day d then: $\sum_{\forall s} f(e,d,s) \leq 1$

Another hard constraint is the minimum and maximum number of employees needed each day to fulfill the cover demand. Given $min_{d,s}$ and $max_{d,s}$, $d \in D$, $s \in S$, there must be: $min_{d,s} \leq \sum_{\forall e} f(e,d,s) \leq max_{d,s}, \forall d, \forall s$.

This kind of constraints may be violated and the program will still be considered feasible. We can divide soft constraints in two categories: The employee soft constraints and the contract soft constraints.

Subsequently, we can categorize the employee soft constraints to: Specific day and shift requirements and pattern requirements.

Let R be the set of the specific day and shift requirements with $R = \{r_1, r_2, ..., r_n\}$, where n is the number of them. Each r_i consists of four variables: $r_i = \{e, d, s, w\}$, where $e \in E$ is the employee that has the requirement, $d \in D$ is the day for the requirement, $s \in S \cup \{A\}$ is the shift type or any shift and $w \in R$ is the weight of the constraint. Then the penalty assigned because of violations is: $penalty_R = \sum_{\forall r_i} (1 - f(e,d,s)) \cdot w)$.

Let M the set of pattern requirements, where $M = \{m_1, m_2, ..., m_n\}$, where n the number of them. Each m_i consists of five variables: $m_i = \{e, min, max, Pat, w\}$ where $e \in E$ is the employee with the requirement, min, $max \in N$ two non negative integer numbers, Pat a sequence of shifts or A, $Pat = \{pt_1, pt_2, ..., pt_l\}$ where l is the pattern length, $pt_i \in S \cup \{A\}$and $w \in R$ is the weigth of the constraint. Then the penalty assigned because of violations is:

$$penalty_M = \begin{cases} w, & \text{if } min > \sum_{j=1}^{p-l+1} k_j \text{ or} \\ & max < \sum_{j=1}^{p-l+1} k_j \\ (\sum_{j=1}^{p-l+1} k_j) \cdot w, & \text{if } min = max = 0 \\ 0, & \text{otherwise} \end{cases}$$

where $k_j = \begin{cases} 1, & \text{if } \sum_{i=0}^{l-1} f(e, d_{j+i}, pt_{i+1}) = l \\ 0, & \text{otherwise} \end{cases}$

Thereafter, we can categorize the contract soft constraints in: Specific pattern requirements, specific day requirements and specific day of the week requirements.

Let Mc the set of contract specific pattern requirements, where $Mc = \{mc_1, mc_2, ..., mc_n\}$, where n the number of them. Each mc_i consists of five variables: $mc_i = \{C, min, max, Pat, w\}$ where $C = e_1, e_2, ..., e_k$ the set of the employees that come under contract C, min, $max \in N$ two non negative integer numbers, Pat a sequence of shifts or A, $Pat = \{pt_1, pt_2, ..., pt_l\}$ where l is the pattern length, $pt_i \in S \cup \{A\}$and $w \in R$ is the weigth of the constraint. Then the penalty assigned because of violations is:

$$penalty_{Mc} = \begin{cases} \sum_C (\sum_{j=1}^{p-l+1} k_j) \cdot w, & \text{if } min = max = 0 \\ \sum_C a_\kappa \cdot w_i, & \text{otherwise} \end{cases}$$

where $k_j = \begin{cases} 1, & \text{if } \sum_{i=0}^{l-1} f(C, d_{j+i}, pt_{i+1}) = l \\ 0, & \text{otherwise} \end{cases}$ and

$$a_\kappa = \begin{cases} 1, & \text{if } min > \sum_{j=1}^{p-l+1} k_j \text{ or } max < \sum_{j=1}^{p-l+1} k_j \\ 0, & \text{otherwise} \end{cases}$$

Let Cd be the set of contract specific day requirements, $Cd = \{cd_1, cd_2, ..., cd_n\}$, where n is the number of them. Each cd_i consists of five variables $cd_i = \{C, d, num, Pat, w\}$ where $C = e_1, e_2, ..., e_k$ the set of the employees that come under the contract c, $d \in D$ the starting day of the requirement, $num \in \{0, 1\}$ a boolean variable that shows if we want existence or absence of the pattern, Pat a sequence of shifts or A, $Pat = \{pt_1, pt_2, ..., pt_l\}$, where l is the patterns length, $pt_i \in S \cup \{A\}$ and $w \in R$ the weight of the constraint. Then the penalty assigned because of violations is:

$penalty_{Cd} = \sum_C a_\kappa \cdot w$ where $a_\kappa = \begin{cases} 0, & \text{if } k_j = num \\ 1, & \text{otherwise} \end{cases}$

and $k_j = \begin{cases} 1, & \text{if } \sum_{i=0}^{l-1} f(C, d_{j+i}, pt_{i+1}) = l \\ 0, & \text{otherwise} \end{cases}$

Let Cw be the set of contract specific day of the week requirements, $Cw = \{cw_1, cw_2, ..., cw_n\}$, where n is the number of them all. Each cw_i consists of six variables $cw_i = \{C, Ds, min, max, Pat, w\}$ where $C = e_1, e_2, ..., e_k$ the set of the employees that come under the contract c, Ds a set of integers that shows which day the pattern is meant to be checked, with $Ds = \{d, d+7, d+2 \cdot 7, ...\}$ and $d \in D$, $min, max \in N$ two non negative integers, Pat a sequence of shifts or A, $Pat = \{pt_1, pt_2, ..., pt_l\}$, where l is the patterns length, $pt_i \in S \cup \{A\}$ and $w \in R$ the weight of the constraint. Then the penalty assigned because of violations is:

$$penalty_{Cw} = \begin{cases} \sum_C (\sum_{Ds} k_{Ds}) \cdot w, & \text{if } min = max = 0 \\ \sum_C a_\kappa \cdot w, & \text{otherwise} \end{cases}$$

$$k_{Ds} = \begin{cases} 1, & \text{if } \sum_{i=0}^{l-1} f(C, d_{Ds+i}, pt_{i+1}) = l \\ 0, & \text{otherwise} \end{cases}$$

$$a_\kappa = \begin{cases} 1, & \text{if } min > \sum_{Ds} k_{Ds} \text{ or } max < \sum_{Ds} k_{Ds} \\ 0, & \text{otherwise} \end{cases}$$

Having defined all the penalties assigned to violations of constraints, we also define F_{V_i}, the sum of them, for each $V_i \in P$, where P is the set of the N in number schedules we are testing, $P = \{V_1, V_2, ..., V_N\}$: $F_{V_i} = penalty_R + penalty_M + penalty_{Mc} + penalty_{Cd} + penalty_{Cw}$.

Finally, we are searching for the program V_i with the lowest F_{V_i}. The objective function is: $\min_{\forall V_i} F_{V_i}$

3 PROPOSED COMPUTATIONAL INTELLIGENCE APPROACHES

In this paper two different computational intelligence approaches are presented: A genetic algorithm (GA) approach and a particle swarm optimization (PSO) approach. Both are a form of heuristic stochastic search that is performed on a population of possible solutions of the problem and it aims to optimize them, based on an objective function.

A GUI is developed, that is divided in two parts: the 'define' part and the 'solve' part. Through the 'define' part the user inserts the number of employees, the period length, the number of contracts and shifts, the personnel requirements for each shift type on each day of the planning period and all types of personal preferences and contract regulations that are described in the above section. In the 'solve' part, the user selects the algorithm type to be executed, along with the parameters that each approach needs. When execution is completed, results are shown in the corresponding fields.

The two approaches work similarly. The first step is to initialize the population, whose size is given by the user through the GUI. Each member of the population is a staff schedule of the problem instance. Each individual is represented by a $n \times p$ matrix, where n is the number of the employees and p the period length. This straightforward representation is chosen because of the simplicity it provides in the constraints' check. The value of each cell is an integer number representing the shift type assigned. The initial solutions are randomly generated so as to fulfill just the requested coverage (hard constraint) and the personal preferences that might exist. An additional consistency check is made, to ensure that assignments are as many as requested by the hard constraint.

Next, each individual is evaluated by the objective function. This function checks for soft constraint violations and sums the penalties, resulting with the fitness of each individual. It returns the fitness along with violation matrices that hold the violations' position on each individual.

After the first evaluation, both algorithms follow an iterative procedure that aims to alter the population, each in a different way and re-evaluate it, until an individual's fitness drops to zero, or if a maximum number of iterations is reached. The last, can be set through the GUI.

3.1 The Genetic Algorithm Approach

The proposed genetic algorithm consists of the same phases as the generic genetic algorithm (Reeves and Rowe, 2003). It begins with the initialization of a population, and loops through evaluation, selection, crossover and mutation operators, until termination criteria are met. An additional phase is added, the consistency check, which is applied after initialization of the population and after the use of the crossover operator, in order to keep the feasibility of the population according to the hard constraint of coverage.

While none of the termination criteria is met, we apply genetic operators on the solutions trying to optimize them. Such operators are selection, crossover, mutation and a neighborhood search.

Two different selection operators are available: roulette wheel selection and tournament selection. The user can select the type that is going to be used, through the GUI. For the second type, the size of the tournaments can also be defined.

The selected chromosomes will be then intersected by one of the four crossover operators that are available. These are crossover by rows, by columns, by rectangles and by the combination of all above. The number of crossover points is random, between one and three points. The probability of appliance can be set through the GUI. Because the crossover operator might produce non feasible solutions, an additional consistency check is made every time it is applied.

After crossover, mutation takes place. There are two types of mutation operators: the inversion of a random number of cells in a column and the targeted mutation of a single gene based on the violation matrix of each chromosome. All parameters can be set through the GUI.

Lastly, a neighborhood search is done by randomly choosing a wrong gene, based on the violation matrix and changing it to all possible values. We evaluate the resulting chromosomes and repeat the above procedure to the chromosome with a better fitness than the initial. The chromosome that is returned is the one with the best fitness from those that occur. This probability can also be set by the user.

As soon as one of the termination criteria is met, the execution breaks and the best chromosome that has been found by the algorithm is returned to the user.

3.2 The Particle Swarm Optimization Approach

Particle Swarm Optimization is a swarm intelligence method based on social behavior. It was developed by James Kennedy (psychologist) and Russel Eberhart (electrical engineer) in 1995 (Kennedy and R.Eberhart, 1995) after studying how a flock of birds moves, influenced by the model of Heppner and Grenander (Heppner and Grenander, 1990). Like genetic algorithms, PSO is a stochastic process, which uses population to find solution within the search space. In PSO the term particles refers to members of the population. The term was inspired by particle physics (Reeves, 1993). Each member of the swarm represents a solution of the problem, which moves in the search space, subject to velocities, looking for the optimal solution.

There are two variants of the basic PSO algorithm, the continuous PSO and the binary PSO. The continuous PSO uses a real valued multidimensional space. In binary PSO the position of each particle is not a real value, but either the binary 0 or 1. Both versions are inappropriate for the problem described above. We propose a discrete PSO with mutation operator (dPSOmo) algorithm that uses the basic idea of the PSO, amended appropriately for the problem.

In the basic PSO each member of the swarm is aware of the best position ever found by a member of the neighborhood it belongs and of the previous best position it has been. In the proposed PSO method, we divide the population into three swarms, each with a different position update rule, with M particles and each particle has knowledge of its current position X, of the best position ever found by any particle of the three swarms *gbest* and of the best position ever found by the particles of the swarm it belongs *lbest*.

After the initialization, the consistency check and the evaluation, the algorithm loops through a three level processing. The first level, mutation, changes a small random number, close to mutation probability set to 0.015, of wrong values in cells of the position of each particle, based on the violation matrix.

The next level is the particles position update. There are three different position update procedures, one for each swarm. The two compute the distances between X and $gbest$ and X and $lbest$. At this point there is a 50% chance of following gbest and another 50% of following $lbest$. If the two positions are very close, the collision is prevented by changing the movement direction. If no collision occurs, the particle moves towards $gbest$ or $lbest$ by copying values of cells from $gbest$ or $lbest$ to X. In the first swarm, this is done by copying whole columns and in the second by copying random individual cells. The third position update procedure follows the velocity update equation: $V = k \cdot (V + f1 \cdot rand \cdot (X - gbest) + f2 \cdot rand \cdot (X - lbest))$ with $k = 0.7398$ and $f1 = f2 = 2.05$. These parameter values were chosen after conducting exhaustive experiments and they seem to yield the best results for all instances. Next, we transform the velocity using the equation: $s = (\frac{2}{1+e^{-|V|}}) - 1$. At this point we create a matrix with random values between 0 and 1, let it be R and the position updates by the following equation:

$$X(R < s) = \begin{cases} lbest(R < s), & \text{if } rand < 0.5 \\ gbest(R < s), & \text{otherwise} \end{cases}$$

The velocity contributes to the determination of the probability that a cell will change to a value from the *gbest* or the *lbest* particle.

Again, if collisions are detected, changes in movement directions are applied.

In order to assure the feasibility of the solutions resulting from the second level of processing, a consistency check is applied in the third level assuring the hard constraint is met.

The algorithm is terminated when the termination criteria discribed earlier are met.

4 EXPERIMENTAL RESULTS

The whole project was implemented in Matlab. The datasets used for testing and comparing the performance of the two algorithms were taken from the ASAP, School Of Science, University of Nottingham website (http://www.cs.nott.ac.uk/ $\sim$ tec/NRP/). The instances of the problem that were selected are the following: Millar-2Shift-DATA1 with 8 employees, 2 shift types, planning period 14 days and cover is per shift, WHPP2 same as WHPP, but divided into two subproblems, one with 20 employees and 2 shift types and the other 10 employees, 1 shift type, both planning period 2 weeks and LLR with 27 employees, 3 shift types, planning period 1 week and cover is per shift.

When using the GA to work on a problem, we must adjust the preferences (population, operator probabilities, etc) according to the requirements. For small sized problems like Millar, having small number of employees and days, the tournament selection works better and elitism is deactivated. This occurs because in tournament selection, the best individual is more likely to be selected for survival in the next generation and in combination with elitism, the possibility to get trapped in a local optima is higher. On the other hand, for larger problems like LLR, the roulette selection and elitism are more appropriate. When using roulette, the selection of the best individual is not as possible as with tournaments, so elitism is needed for the algorithm to converge. Moreover, if we apply more operators, with smaller probabilities, the results further improve.

On the contrary, PSO uses the same parameter values (k, f1, f2, mutation probability) for all problem instances. These parameter values yield the best results for all instances after conducting exhaustive results.

Both algorithms are executed for the same number of iterations, 100 monte carlo runs for each problem. The population number differs; for GA is set to 50 chromosomes and 150 for the Millar dataset and for PSO is set to 15 particles for all datasets. The results are shown in tables 1 and 2.

If we increase the number of iterations, then the results of LLR further improve.

Table 1: Results in 10000 iterations.

Problem	Method	Average	Best	Ideal
Millar	GA	183.33	0	0
	dPSOmo	116,66	0	
LLR	GA	363.5	342	301
	dPSOmo	319.33	306	
WHPP2	GA	220.166	85	5
	dPSOmo	47	43	

Table 2: Results in 60000 iterations.

Problem	Method	Average	Best	Ideal
LLR	GA	350,66	326	301
	dPSOmo	315,83	303	
WHPP2	GA	51.5	36	5
	dPSOmo	29.5	23	

5 CONCLUSIONS

In the present paper we defined the staff scheduling problem as a shift scheduling problem with personal and contract constraints, we proposed a mathematical model for the problem and implemented two algorithms to solve it. A parameterized genetic algorithm and a discrete particle swarm optimization with mutation operator algorithm were tested in three instances of the problem.

Both methods seem to be effective, with a clear lead of the dPSOmo method. The Genetic Algorithm approach has too many parameters and its performance may be improved with a more thorough experimentation on the parameters value. On the other hand dPSOmo does not need parameter adjustment and performs well in every instance. There are few existing approaches to solve the staff scheduling problem with PSO (Nissen and Günther, 2009; Günther and Nissen, 2009), most of them solving the subdaily scheduling with workstation and the comparison of the approaches would be inaccurate, because of the different problem formulation. In our approach for the shift staff scheduling, we developed a novel PSO variation, that divides the population into three swarms, each following different update rules. Also, mutation is applied to the cells that conflict with a constraint, based on the violation matrix.

The experimental results are very promising and the PSO variation has been proved to outperform Genetic Algorithms which is one of the state-of-the-art solutions of the staff scheduling problem.

Our future plans involve the application of adaptive parameter values on both proposed methods in order to encourage global search for the initial algorithms‘ generations and local search for the final generations. Furthermore, these meta-heuristic methods will be integrated and hybridized with accurate local search approaches in order to increase the accuracy and the convergence velocity. Finally, the proposed computational intelligence techniques will be applied in real life data in order to measure their performance in even harder staff scheduling problems.

REFERENCES

Baker, A., Roach, G., Ferguson, S., and Dawson, D. (2003). The impact of different rosters on employee work and non-work time preferences. *Time Society*, 12:315–332.

Bartholdi, J. J., Orlin, J. B., and Ratliff, H. D. (1980). Cyclic scheduling via integer programs with circular ones. *Operations Research*, 28:1074–1085.

Brusco, M. (1995). Cost analysis of alternative formulations for personnel scheduling in continuously operating organizations. *European Journal of Operational Research*, 86:249–261.

Burke, E. K., Causmaecker, P. D., and Landeghem, H. V. (2004). The state of the art of nurse rostering. *Journal of Scheduling*, 7:441–499.

Burke, E. K., Curtois, T., Post, G. F., Qu, R., and Veltman, B. (2008). A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. *European Journal of Operational Research*, 188:330–341.

Cheang, B., Li, H., Lim, A., and Rodrigues, B. (2003). Nurse rostering problems - a bibliographic survey. *European Journal of Operational Research*, 151:447–460.

Chen, J. and Yeung, T. (1992). Development of a hybrid expert system for nurse shift scheduling. *International Journal of Industrial Ergonomics*, 9:315–327.

Dowsland, K. (1998). Nurse scheduling with tabu search and strategic oscillation. *European Journal of Operational Research*, 106:393–407.

Easton, F. F. and Mansour, N. (1993). A distributed genetic algorithm for employee staffing and scheduling problems. In *International Conference on Genetic Algorithms*, pages 360–367.

Günther, M. and Nissen, V. (2009). A comparison of neighbourhood topologies for staff scheduling with particle swarm optimisation. In *German Conference on Artificial Intelligence*, pages 185–192.

Heppner, F. and Grenander, U. (1990). A stochastic nonlinear model for coordinated bird flocks.

Jaumard, B., Semet, F., and Vovor, T. (1998). A generalized linear programming model for nurse scheduling. *European Journal of Operational Research*, 107:1–18.

Kennedy, J. and R.Eberhart (1995). *Particle Swarm Optimization*, volume IV, pages 1942–1948.

Millar, H. and Kiragu, M. (1998). Cyclic and non-cyclic scheduling of 12 h shift nurses by network programming. *European Journal of Operational Research*, 104:582–592.

Nissen, V. and Günther, M. (2009). Staff scheduling with particle swarm optimisation and evolution strategies. In *EvoWorkshops*, pages 228–239.

Özcan, E. (2005). Memetic algorithms for nurse rostering. In *International Symposium on Computer and Information Sciences*, pages 482–492.

Ozkarahan, I. and Bailey, J. E. (1988). Goal programming model subsystem of a flexible nurse scheduling support system. *Iie Transactions*, 20:306–316.

Reeves, C. and Rowe, J. (2003). Genetic algorithms: Principles and perspectives: A guide to ga theory.

Reeves, W. (1993). Particle systems-a technique for modelling a class of fuzzy objects. *Computers & Graphics*.

Weil, G., Heus, K., Francois, P., and Poujade, M. (1995). Constraint programming for nurse scheduling.