Cooperative Search for Fair Nurse Rosters

Simon Martin[1], Djamila Ouelhadj[1], Pieter Smet[2], Greet Vanden Berghe[2], Ender Özcan[3]

[1] *Logistics and Mathematics Management Group, University of Portsmouth, Department of Mathematics, UK*
email: simon.martin@port.ac.uk, djamila.ouelhadj@port.ac.uk
[2] *CODeS, KAHO, Computer Science, Belgium*
ITEC, iMinds, KULeuven, Belgium.
e-mail: pieter.smet@kahosl.be, greet.vandenberghe@kahosl.be
[3] *Automated Scheduling, Optimisation and Planning Research Group, University of Nottingham, Department of Computer Science, UK*
email: ender.ozcan@nottingham.ac.uk

Abstract

The development of decision support systems acceptable for nurse rostering practitioners still presents a daunting challenge. Building on an existing nurse rostering problem, a set of fairness-based objective functions recently introduced in the literature has been extended. To this end, a generic agent-based cooperative search framework utilising new mechanisms is described, aiming to combine the strengths of multiple metaheuristics. These different metaheuristics represent individual planners' implicit procedures for improving rosters. The framework enables to explore different ways of assessing nurse rosters in terms of fairness objectives. Computational experiments have been conducted across a set of benchmark instances. The overall results indicate that the proposed cooperative search for fair nurse rosters outperforms each metaheuristic run individually.

Keywords: Cooperative search, agent-based systems, nurse rostering, fairness.

1. Introduction

Optimisation problems are usually solved by one or more human planners who developed a mental model of the objective to be optimised and a heuristic procedure for generating solutions. One of the hardest issues to address when developing decision support systems is modelling of the problem in a way that the human planners experience as natural, while producing solutions that reveal part of the optimisation process. When multiple decision makers are involved, not necessarily sharing the same models and the same heuristics, the problem turns out to be even more challenging. An obvious example can be found in hospitals, where individual nurses cooperate and have a say in establishing the monthly roster for the entire ward, at least when the process is not fully automatised. An interesting question is whether situations with multiple decision makers can be mimicked into an optimisation approach and whether the obtained results are better than after optimisation from a single planner's point of view.

Nurse rostering is a domain of interest to many researchers and practitioners across different fields, including computer science, operational research and artificial intelligence (Burke et al., 2004, 2001; Beddoe and Petrovic, 2006; Özcan, 2005, 2007; Petrovic and Vanden Berghe, 2012). A *nurse roster* is a timetable consisting of shift assignments of nurses at a health-care institution. Creating nurse rosters subject to a set of constraints based on individual preferences and on requirements imposed by administration is a challenging task (Even et al., 1976; Karp, 1972).

Cooperative search is defined as a search process performed by agents capable of exchanging information, e.g., models, solutions and/or other search space characteristics during the search process (Clearwater et al., 1992; Crainic and Toulouse, 2008; Blum and Roli, 2003; Hogg and Williams, 1993; Toulouse et al., 1999). It has been successfully used to solve a number of difficult combinatorial optimisation problems, such as multi-commodity location with balancing requirements (Crainic et al., 1995, 1997), capacitated network design (Crainic and Gendreau, 2002), vehicle routing (Bouthillier and Crainic, 2005), quadratic assignment (James et al., 2009), labour constraint scheduling (Cavalcante et al., 2001), permutation flow shop scheduling (Ouelhadj and Petrovic, 2010; Vallada and Ruiz, 2009). These studies have shown that the combination of several metaheuristics with different parameter settings increases the robustness of the global search relative to variations in problem instance characteristics.

Agent-based cooperative metaheuristic approaches are fairly new in nurse rostering. Wang and Wang (2009) developed an agent-based approach to self rostering. Haspeslagh et al. (2009) addressed the problem of exchanging nurses between wards and sorting out personnel shortages. Haspeslagh et al. (2009) presented a Pareto optimal negotiation approach for leveraging the workload across different wards. Most of the work on agent-based fairness centres around game theoretic approaches (De Jong et al., 2008) where the agents are used to model human behaviour and interactions. To the best of our knowledge, there has been no work in the literature on agent-based cooperative search for fairness making use of multiple fairness-based objective functions.

Smet et al. (2013b) introduced different different fairness-based objective functions representing individual planners' impressions of good quality rosters We propose a novel cooperative search agent-based framework that uses this recent work. (Smet et al., 2013a) present different solution methods which may correspond to the planners' individual and implicit procedures for improving rosters. The cooperative search agent-based framework combines the strengths of different metaheuristics and fairness models. The agents are able to use the same or different fairness objective functions enabling them to find the most appropriate model for a given problem. By the same token, each agent in this framework is autonomous and capable of executing different metaheuristic combinations with different parameter settings. The agents cooperate asynchronously using pattern matching and reinforcement learning to produce high quality solutions with respect to fairness.

A brief overview of the nurse rostering problem and a set of alternative fairness objective functions along with measures of fairness for nurse rostering are provided in Section 2. In Section 3, we present an agent-based framework for cooperative metaheuristic search for fair nurse rostering. The experimental design is discussed in Section 4, including the implementation details of different metaheuristic agents. The results of the computational experiments are reported in Section 5. Section 6 concludes with a discussion on the contribution and some directions for future research.

2. Problem description and fairness-based objective functions

The goal of automated nurse rostering is to assign shifts to a set of nurses so that 1) the minimum staff requirements are fulfilled and 2) the nurses' contracts are respected (Burke et al., 2004).

Nurse rostering is a highly constrained scheduling problem which was proven to be NP-hard (Karp, 1972) in its simplified form. The problem is represented as a constraint optimisation problem using 5-tuples: (i) set of nurses, (ii) set of days (periods) including the relevant bits from the previous

Hard constraints	Soft constraints
Single Assignment Start Per Nurse Per Day	Coverage Constraints
No Overlap between Assignments	Assignment to the Primary Skill
Honour Skill Types	Rest Times
Operations on Defined Assignments Only	Requests
Schedule Locks	Time-related Constraints

Table 1: Hard and soft constraints

and upcoming schedule, (iii) set of shift types, (iv) set of skill types and (v) constraints.

The nurse rostering model addressed in this paper is based on the model of Bilgin et al. (2012). A roster consists of a set of assignments from the discrete space defined by nurses, skill types, shift types, and schedule period subject to given constraints. It defines different types of constraints: hard, soft, time-related and coverage constraints. The hard and soft constraints of the model are described in Table 1 and a full description of this model can be found in Bilgin et al. (2012); Smet et al. (2013a). A roster is *feasible* if and only if all hard constraints are satisfied. Times constraints are a type of soft constraint concerned with sequences of shift patterns. Soft constraints represent preferences. A solution method aims at resolving as many soft constraints as possible for a given problem. When a soft constraint is violated, a penalty is incurred proportional to the importance of the constraint and the severity of its violation. The importance of each constraint is expressed using weights. The higher the weight, the more important the constraint is, relative to the other constraints. A frequently used objective function in nurse rostering is referred to as the weighted sum objective function $MinWS$. Let $c \in C$ be the set of soft constraints then, w_c is the weight associated with constraint c and n_c the number of violations of c. We define:

$$q_{ros}(i) = \sum_{0 \leq c \leq |C|} w_c n_c \tag{1}$$

We also define p_{viol} which represents the coverage violations accrued by any over- or under-staffing in a given roster. Therefore $MinWS$ consists of two parts: $q_{ros}(i)$ associated with the costs of assigning a nurse to a given shift and p_{viol} the coverage constraints. Thus the objective is to find a roster with the lowest overall penalty, denoted as $MinWS$.

$$MinWS = \min\left(\sum_{n\in N} q_{ros}(n) + p_{viol}\right) \tag{2}$$

Smet et al. (2013b) address the issue of fairness in automated nurse rostering. The fair distribution of contractual violations among nurses is an important contributor to the satisfaction nurses get based on their work schedules, and thus a large influence on the overall job satisfaction of nurses. Smet et al. (2013b) show that, the commonly used $MinWS$ does not result in fair solutions. They investigated alternative fairness models from the literature (Julian et al., 2002; Vasupongayya and Chiang, 2005) to solve the problem of fairness in nurse rostering and show that by optimising these models, fairer solutions are obtained, generally at the cost of an increased number of constraint violations. The objective functions they introduced will also be used in this study. They are defined below by Equations 4, 5 and 6.

All these fairness objective functions try to even the distribution of nurse roster violations so that no nurse is favoured over another.

First of all, we define the average individual roster quality of the nurses: μ, calculated as:

$$\mu = \frac{1}{|N|}\sum_{n\in N} q_{ros}(n) \tag{3}$$

Based on (Smet et al., 2013b) we describe the four fairness objective functions used in this study:

$$MinMax\ = \min\left(|N|\left(\max_{n\in N} q_{ros}(n)\right) + p_{viol}\right) \tag{4}$$

$$MinDev = \min\left(\left(\sum_{n\in N}(|\mu - q_{ros}(n)|) + |N|\,\mu\right) + p_{viol}\right) \tag{5}$$

$$MinError = \min\left(|N|\left(\left(\max_{n\in N} q_{ros}(n) - \min_{n\in N} q_{ros}(n)\right) + \mu\right) + p_{viol}\right) \tag{6}$$

We also introduce the objective function $MinSS$ which is similar to $MinMax$ as it tries to reduce the worst roster of a given nurse. However, it does this by emphasising on the worst individual rosters by squaring. The aim is therefore to reduce these poor individual rosters. w_1 and w_2 are weights used to scale the two parts of equation.

$$MinSS = \min \left(w_1 \sum_{n \in N} q_{ros}(n)^2 + w_2 p_{viol}^2 \right) \tag{7}$$

3. Agent-based cooperative metaheuristic search

3.1. An agent-based framework for cooperative metaheuristic search

Cooperative search provides a class of strategies to design more effective search methodologies by combining metaheuristics for solving combinatorial optimisation problems. This area has been little explored in operational research. We introduce a generic agent-based distributed framework where each agent implements a metaheuristic and a fairness objective function. An agent continuously adapts itself during the search process using a cooperation protocol based on reinforcement learning and pattern matching. Good patterns which make up improving solutions, are identified and shared by the agents.

The agent-based system is composed of a launcher agent and metaheuristic agents (Martin et al., 2012).

- **Launcher agent**: The launcher agent reads problem instances from XML benchmark files, executes seed algorithms using one of the fairness objective functions and then sends the seed schedules, one at a time, to each of the metaheuristic agents. After the improving phase, it also collects the final schedules from the metaheuristic agents and selects the fairest rosters.

- **Metaheuristic agent**: The metaheuristic agents receive a problem from the the launcher agent which they try to optimise using their given metaheuristic and local search heuristic combinations. They also cooperate asynchronously with the other agents according to a cooperation protocol where the agents try to find recurring good schedule patterns. The agents share the good patterns and each try to build new improving rosters which they then optimise according to the given fairness objective functions and metaheuristics. After a number of iterations, the metaheuristic agents each send their best roster to the launcher. The launcher then chooses the fairest roster.

The metaheuristic agents cooperate asynchronously by iterating a communication protocol which is a distributed algorithm. At each iteration a conversation initiator (identified in the previous iteration) sends the result of its latest metaheuristic search using its own fairness objective function to the other agents. The agents compare this result with their own and then generate patterns (Section 3.3) which they send back to the initiator. The initiator then identifies good patterns and shares them amongst the metaheuristic agents. Each agent then generates a new potential solution. The metaheuristic agent with the lowest objective function value in the present iteration is chosen to be the initiator of the next. In the next iteration the new potential solutions are used to further the search in the manner just described.

Figure 1 shows how the agents perform a search by cooperating with each other running asynchronously in parallel and executing different metaheuristic and heuristic combinations with different parameter settings. They use ontologies (Section 3.2) to enable the agent-based system to be adapted easily to new problem domains.

The internal structure of a metaheuristic agent is mediated at two levels through the use of ontologies. Ontologies are defined as a set of general representational primitives to model semantic elements of a domain (Gruber, 1993).

3.2. Ontologies

The ontology currently used by the framework generalises the notions of: *assignments*, *pairs of assignments*, *constraints* and *roster or schedule.* In the ontology these are called *SolutionElements, HeuristicData, Constraints* and *SolutionData* objects respectively.

- **SolutionElements:** A SolutionElement represents the assignment of a nurse with specific skills to a shift on a certain day as required by a given nurse rostering problem.

- **HeuristicData:** A HeuristicData object contains two SolutionElements objects. These represent pairs of *assignments* in a roster that

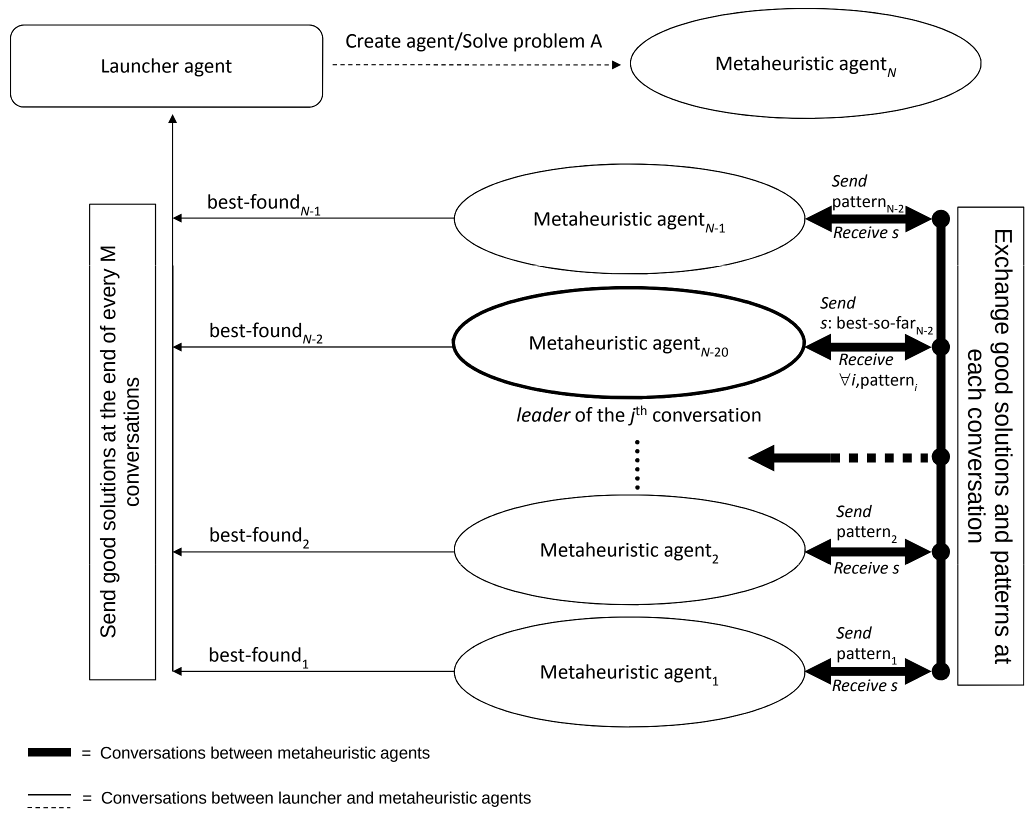


Figure 1: Agent-based framework for cooperative metaheuristic search

will be used in the cooperation protocol to identify good patterns of assignments in improving rosters.

- **Constraints:** The Constraints interface is the interface between the high level framework and the concrete constraints used by the nurse rostering problem. These are used to verify a valid roster.

- **SolutionData:** A SolutionData object represents the list of assignments that make up a valid nurse roster.

The ontology is represented in XML, corresponding to the representation of most of the nurse rostering benchmark problems. This makes the interface between problem definition and ontology seamless in practice. As a consequence the elements defined in a specific problem instance will be used directly by the system. Figure 2 shows the structure of the ontology and how SolutionElements is the interface between the framework and the nurse rostering problem instances.

The SolutionElements, HeuristicData, Constraints and SolutionData objects are also used in the agents to facilitate inter-agent cooperation.

The next section describes the cooperation protocol and how this is implemented with the use of ontologies.

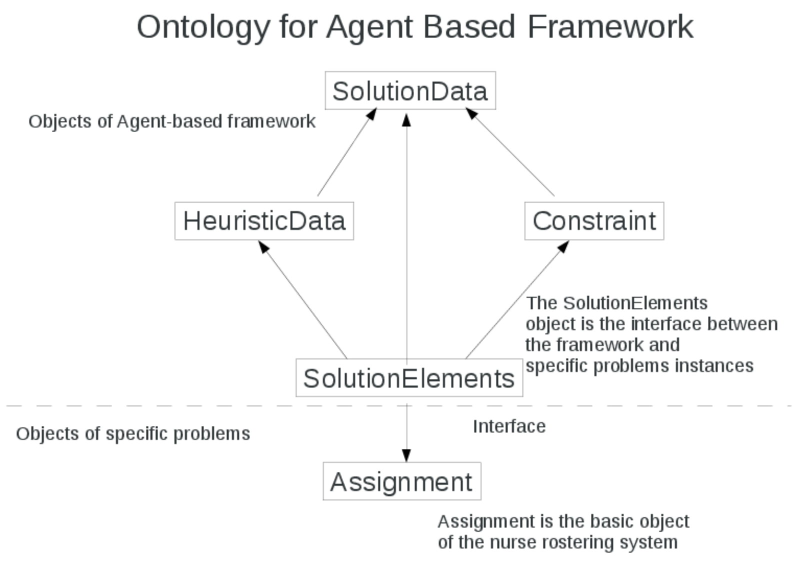

Ontology for Agent Based Framework
SolutionData
Objects of Agent-based framework
HeuristicData
Constraint
The SolutionElements object is the interface between the framework and specific problems instances
SolutionElements
Objects of specific problems
Interface
Assignment
Assignment is the basic object of the nurse rostering system

Figure 2: The interface between the framework and problem instances

3.3. Asynchronous cooperative search by pattern matching and reinforcement learning for nurse rostering

The agents cooperate by exchanging patterns defined as a permutation of assignments. This is a simple distributed metaheuristic.

Given such a permutation of assignments, it is always possible to generate n pairs from it. In the following example, we define the permutation of nurse assignments using the following ID's, where $n = 10$: <2,4,7,6,5,8,9,0,1,3>. The following n pairs can be generated:

$$(2,4),(4,7),(7,6),(6,5),(5,8),(8,9),(9,0),(0,1),(1,3),(3,2)$$

.

The permutation is broken into patterns of the same length while retaining the basic order of the permutation. The agents can then each compare pairs of assignments generated by their own metaheuristic with pairs shared by the other agents. All the pairs from each agent are scored based on how frequently they appear. Only those pairs that have the highest frequency score are shared out amongst the agents. The idea behind this step is that the lists of assignments have already been optimised by each agent's respective metaheuristic so each list must have elements that make up a good roster. It is argued that these pairs will occur more frequently as the search progresses.

Once the good pairs are shared out, the agents use these pairs to generate a new roster by using a greedy heuristic. This is a simple heuristic that makes the new roster by taking the good patterns first and then filling the rest of the roster with the agent's previous best-so-far permutation. The new permutation of assignments is now ready to be optimised by an agent's metaheuristic and fairness objective function.

This protocol involves diversification and intensification phases. Diversification involves pattern matching and the application of the greedy heuristic to produce a new roster. This roster is then intensified using a metaheuristic. One application of these two phases is called a *conversation*. It should be noted that the pattern matching results in a controlled diversification of the search. Good patterns are retained and used to build new potential solutions which will diversify the search just enough to enable new areas of the solution space to be explored. The metaheuristics then intensify the search.

The asynchronous cooperative search is implemented as follows:

1. A metaheuristic agent taking on the role of conversation initiator starts a conversation. It takes a new roster either generated from a previous conversation or supplied by the launcher agent. The new roster is then improved by the initiator agent. When an improved roster is generated locally, it is sent to the other metaheuristic agents.
2. The metaheuristic agents have also generated their best-so-far rosters using their fairness objective functions. They break up the rosters sent from the initiator and their own into pairs. The pairs are then compared and only those that are common to both rosters are kept. HeuristicData objects are created from these pairs storing the first and second assignments of the pair. These are then sent by the metaheuris-

tic agents to the initiator. The metaheuristic agents also send the value of their fairest roster found so far. These rosters will be used by the initiator to determine which metaheuristic agent will be the new initiator in the next conversation.

3. Upon receiving the HeuristicData objects from the other metaheuristic agents, the initiator pools them locally. Each HeuristicData object is scored by counting how frequently it occurs in the pool. The initiator then tries to build a linked list from these high scoring HeuristicData objects.
   For example, if the pool contains the following HeuristicData objects with first and second assignments expressed here as pairs (4,7) (6,1) (7,2) (2,6) (5,9) (3,8), the linked list generated from the HeuristicData objects will have the following order (4,7) (7,2) (2,6) (6,1). Any HeuristicData objects not linked in this way are stored in an unlinked list (5,9) (3,8).
4. The initiator then determines which metaheuristic agent is going to be the initiator in the next conversation. This is done by pooling all the fairness objective function values of the best rosters found so far by the metaheuristic agents and then identifying which metaheuristic agent has the best fairness objective function value. The metaheuristic agent with the best objective value will be the new initiator in the next conversation. The initiator then sends these lists of best-so-far HeuristicData objects to the metaheuristic agents. In the same message it also indicates which metaheuristic agent will be the new initiator in the next conversation.
5. The metaheuristic agents receive the lists of HeuristicData objects. Both initiator and metaheuristic agents then create a new roster from these objects, as well as their current best roster. The new roster is created by trying to build first a list of HeuristicData objects, while the metaheuristic agent's best-so-far roster provides any missing numbers. In this way a new unique roster for each agent is generated and the objective function value is calculated.
6. The conversations are repeatedly exchanged between the metaheuristic agents for a maximum number of conversations set in the configuration file of the launcher agent.

4. Experimental Design

Section 4.1 discusses the configuration of the agent-based system. Section 4.2 real-world examples from two Belgian hospitals are described that are used in the experiments. We also describe two different scenarios and the measures used to verify the fairness of a roster.

4.1. Implementation of the agent-based framework

The agent-based framework for cooperative search is implemented using the open source FIPA compliant development platform JADE (Bellifemine et al., 2005). We implemented three different metaheuristic agents in the framework:

- **Tabu Search agents (Glover, 1990)**: The tabu search agents implement a basic tabu search. The search starts from a feasible roster and moves iteratively from the current roster to its best neighbouring roster using neighbourhoods even if that neighbourhood worsens the objective function value.
- **Simulated Annealing agents (Bertsimas and Tsitsiklis, 1993)**: The simulated annealing agents implement the basic algorithm with a geometric cooling schedule.
- **Variable Neighbourhood Search agents (Bilgin et al., 2012)**: The variable neighbourhood search agents implement a perturbation heuristic which uses local search heuristics to improve an initial solution. The algorithm proceeds by randomly choosing a heuristic from the list of available neighbourhood operators. Each possible move is validated against the hard constraints ensuring that at every stage of the search each new solution remains feasible.

Tabu Search, Simulated Annealing and VNS use the local search moves developed by Bilgin et al. (2012). The neighbourhoods are:

- **Shift**: It assigns a nurse to a new shift to the roster.
- **Delete shift**: This move deletes a shift from the roster.
- **Single shift day**: An assignment is removed from a nurse's schedule and added to another nurse on the same day if the second nurse has no assignment on that day and has the associated skill type.

- **Change assignment based on compatible shift type**: The shift type of an assignment is changed to another compatible shift type defined in the coverage constraints for the associated day and skill type.
- **Change assignment based on skill type**: This neighbourhood can be used when the nurses have at least two different skill types. It deletes an assignment and adds another assignment to one of the nurse's other skill types.
- **General Assignment Change**: An assignment is changed to another shift type where the skill type of the assignment remains the same.

The experiments were conducted using a network of 13 agents, configured as follows:

- Launcher agent
- 4 Tabu Search agents
- 4 Simulated Annealing agents
- 4 Variable Neighbourhood Search agents

The metaheuristic agents are configured from a file which they execute at start-up. This file specifies to the agent which metaheuristic to run; how many iterations to perform; which fairness objective function to use; also, parameters such as the size of tabu tenure or which cooling schedule function to use if configured to run SA. All agents run the same local search heuristics described above.

The parameter settings of the individual agents remain unchanged throughout all testing. Each agent evaluates its given metaheuristic for 500 iterations or until no further improving solutions are found. Experimentation shows that it provides a good balance between diversification using pattern matching while the 500 iterations of the metaheuristic intensifies the new roster just enough that the agents tend to converge to good rosters. TS agents have a tabu tenure set to seven. SA agents use a geometric temperature cooling schedule set at 0.9 for a more diverse search. All agents randomly select the next local search heuristic to run.

Each agent ran on a HP Compaq 6000 pro with Intel Core 2 duo E8400 processor with 4 GB RAM in a 2x2 GB configuration. The agents were configured to use only 1 GB of memory.

4.2. Experimental data and fairness evaluation measures

The experiments were conducted on the data used by Smet et al. (2013b)[1]. The instances are based on four different hospital wards from two Belgian hospitals: emergency, geriatrics, psychiatry and reception. Table 2 provides an overview of the instance characteristics. For each ward, two cases are considered: one where all the nurses have the same contract (referred to as name of instance_i), and one where each nurse has an individual contract with both common and personalised constraints (referred to as name of instance_d).

Instance	No. nurses	No. shifts	No. skills	Planning period
Emergency	27	27	4	28 days
Geriatrics	21	9	2	28 days
Psychiatry	19	14	3	31 days
Reception	19	19	4	42 days

Table 2: The properties of the benchmark instances used during the experiments obtained from a Belgian hospital.

The fairness objective functions guiding the search process towards fair rosters and the metrics measuring fairness are separated. This is due to the reason that the objective functions take into account different components, particularly, contractual constraints and coverage constraints during the search process, while the fairness metrics ignore the coverage constraints to better assess how fair the full roster is from the point of view of nurses. In this study, we use the Jains fairness index (Jain et al., 1984; Muhlenthaler and Wanka, 2012), denoted by $Jains$, to measure the fairness of a roster. Equation 8 shows how this measure is calculated for a given solution with N, the set of nurses.

$$Jains = \frac{\left(\sum\limits_{n \in N} q_{ros}(n)\right)^2}{|N| \cdot \sum\limits_{n \in N} q_{ros}^2(n)} \tag{8}$$

[1]`http://allserv.kahosl.be/~pieter/nurserostering`

The value of Jains fairness index varies from $1/|N|$ (worst fairness case) to 1 (best fairness case), which denotes that the violations for each individual nurse is the same. If the assignment of a set of nurses to form a roster generates no violations then the value of the roster is 1. In such as case the roster is deemed to be completely fair.

To compare the performance of two different fairness objective functions, A and B over a given set of instances, we use %gap indicating the percentage change that an algorithm using the objective function B generates over A (i.e., gap from A to B):

$$\%gap = \frac{A - B}{B} \times 100 \tag{9}$$

Assuming that $A = MinFO$, where $MinFO \in \{MinMax$, $MinDev$, $MinError$, $MinSS\}$ and $B = MinWS$, if an algorithm using the minimising objective function $MinFO$ improves the (average) solution quality when compared to an algorithm using $MinWS$ over a set of runs for an instance, then (average) $\%gap < 0$. If B is 0 the search has found a solution with no violations. This $\%gap$ measure is not applicable (N/A) in this case.

4.3. Experimental Scenarios

Two experimental scenarios were conducted to evaluate the performance of the proposed cooperative search. In the first scenario, all 12 metaheuristic agents executed the same objective function on all the instances. Each objective function was tested separately in this manner. The same tests were conducted in stand alone mode where metaheuristic agents do not cooperate. Here a single agent runs the same experiments evaluating its fairness objective function the same number of times as all the 12 cooperating agents added together. For example, in cooperation mode 12 agents conducted 200 conversations evaluating their objective functions as most 500 times. Therefore, in the stand alone experiments, a metaheuristic evaluates the fairness objective function $12 \times 200 \times 500 = 1200000$ times. In all the experiments, $w_1 = w_2 = 1$ for $MinSS$.

The Jains fairness index defined in (Equation 8) is used to evaluate the relative fairness of the solutions produced by each configuration. The average percentage increase (Equation 9) from $MinWS$ for each instance was used to evaluate the efficiency of each fairness objective function in terms of number of constraint violations.

In the second scenario, the same instances were used but all four fairness objective functions are used at the same time to generate the nurse rosters. The idea behind this test is to identify which fairness measures are best for each problem instance. Of the twelve metaheuristic agents, a group of three different metaheuristic agents generated nurse rosters using $MinMax$, three ran $MinDev$, three ran $MinError$ and finally three ran $MinSS$. When the search is finished, each agent sends their best roster to the launcher which then selects the roster with the largest $Jains$ fairness measure.

All experiments were conducted on real-world benchmark problems with 20 repeated runs for each problem instance. In the first scenario, these experiments were conducted five times for each of the five objective functions. The resulting nurse rosters were then each recalculated using the other objective functions not used in the optimisation of the roster so that they could be compared with the original. In the second scenario each experiment was also repeated 20 times for each instance. The agents conducted only 200 conversations to complete each search taking no longer than 6 minutes to produce a new roster for each problem instance.

5. Computational Results

In this section, we present the results of the two test scenarios described in Section 4.3. In the first test scenario, the metaheuristic agents use cooperative search to generate the nurse rosters using the same fairness objective function, while in the second one, the metaheuristic agents use cooperative search and a mixture of the fairness objective functions to generate the nurse rosters. As a statistical test, a Wilcoxon singed-rank test has been performed to compare pairwise performance of two given algorithms. The following notation is used: Given A versus B, $<$ ($\leq$) denotes that B is better than A and this performance variance is (not) statistically significant within a 95% confidence level. The software used for generating and evaluating rosters was provided by the authors of (Smet et al., 2013a)[2].

5.1. Scenario 1 test results

Table 3 summarises the first scenario results using average Jains fairness index over 20 runs for each instance. Considering the average performance

[2]Personnel rostering software kernel, KAHO, Gent

of cooperative search with respect to the Jains fairness index using different fairness objective functions, $MinDev$ ad $MinMax$ are the best choices under scenario 1, performing well on all instances. Their performance is significantly better than the rest of the objective functions. $MinDev$ performs better than $MinMax$ on average across five instances including Emergency_i, Emergency_d, Geriatric_i, Geriatric_d, Psychiatry_d and this performance difference is statistically significant. $MinDev$ and $MinMax$ both deliver a similar performance on Psychiatry_i. On the other hand, $MinMax$ performs significantly better than $MinDev$ on the Reception instances. $MinError$, $MinSS$ and $MinWS$ follows $MinMax$ in the given order when their overall average performances are compared. Smet et al. (2013b) showed that $MinWS$ produces the least fair rosters and similar phenomena are observed under the cooperative search Scenario 1 setting.

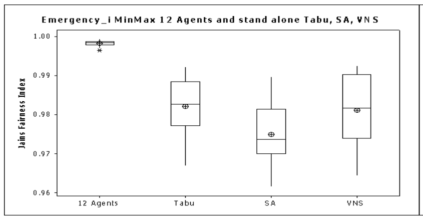
Instance	MinWS	MinMax	MinDev	MinError	MinSS
Emergency_i	0.5638	0.9983	**0.9991**	0.9074	0.9609
Geriatric_i	0.4451	0.9784	**0.9861**	0.6023	0.5613
Psychiatry_i	0.6069	**0.9995**	**0.9995**	0.9054	0.8938
Reception_i	0.6273	**0.8808**	0.8515	0.7191	0.1076
Emergency_d	0.6295	0.9973	**0.9982**	0.8574	0.9228
Geriatric_d	0.6013	0.9980	**0.9986**	0.9491	0.3349
Psychiatry_d	0.4446	0.9986	**0.9988**	0.6882	0.6719
Reception_d	0.3321	**0.9895**	0.9824	0.8052	0.2603

Table 3: The average Jains fairness index over 20 runs of a given fairness-based objective function for each benchmark instance under Scenario 1, where the bold entries indicate the best one for a given instance.

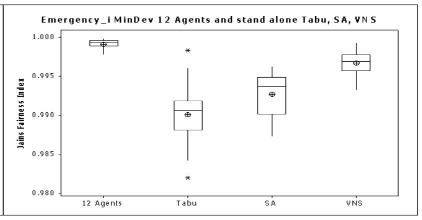
Table 4 shows the average Jains fairness index over 20 runs for each instance considering the stand-alone tests in which a single metaheuristic is used as a sequential metaheuristic. When a given metaheuristic makes use of a different objective function, its performance changes. In general, $MinDev$ is a better choice than the other fairness-based objective functions to be used by a stand-alone metaheuristic. $MinMax$ is the second best choice as an objective function to produce fair rosters. As expected, $MinWS$ does not generate high quality solutions as well as the other objective functions in terms of fairness in most of the cases. $MinWS$ generates fairer solutions when compared to $MinSS$ for Reception_i, regardless of the standalone algorithm used. A similar phenomenon is observed for the standalone VNS using $MinWS$ when compared to VNS using $MinWS$ on the Geriatric instances. VNS performs slightly better than the other stand-alone metaheuristics in the overall when $MinDev$ is used as the fairness-based objective function. Cooperative search based on Scenario 1 using $MinDev$ performs significantly better than all stand-alone metaheuristics regardless of the objective function used on all instances, except for Psychiatry_i and Reception_i. For Psychiatry_i, Scenario 1 using $MinDev$ performs similar to the stand-alone VNS using $MinDev$. For Reception_i, stand-alone VNS using $MinMax$ performs better than Scenario 1 using $MinDev$. Figure 3 provides the box plots of Jains fairness index for the Emergency_i instance over 20 runs considering cooperative and stand-alone metaheuristics using different objective functions as an example. The results indicate the success of cooperative fairness approach over the approach without cooperation when $MinDev$, $MinMax$ or $MinSS$ is used as an objective function. This performance variation is statistically significant.

Stand-alone VNS					
Instance	*MinWS*	*MinMax*	*MinDev*	*MinError*	*MinSS*
Emergency_i	0.7477	0.9812	**0.9966**	0.8697	0.9605
Geriatric_i	0.5646	**0.9714**	0.9646	0.5413	0.6101
Psychiatry_i	0.8070	0.9874	**0.9995**	0.9652	0.9809
Reception_i	0.7148	**0.8893**	0.8277	0.8192	0.5766
Emergency_d	0.6829	0.9845	**0.9955**	0.8094	0.9556
Geriatric_d	0.6362	0.9747	**0.9981**	0.9466	0.4337
Psychiatry_d	0.6299	0.8929	**0.9931**	0.7714	0.9158
Reception_d	0.4970	0.8654	0.7106	0.6601	0.5618
Stand-alone Simulated Annealing					
Instance	*MinWS*	*MinMax*	*MinDev*	*MinError*	*MinSS*
Emergency_i	0.8657	0.9751	**0.9927**	0.9337	0.9314
Geriatric_i	0.6789	**0.9718**	0.9533	0.8117	0.7530
Psychiatry_i	0.8979	0.9542	**0.9929**	0.9532	0.9809
Reception_i	0.7660	**0.8380**	0.8332	0.8323	0.7792
Emergency_d	0.8222	0.9592	**0.9706**	0.8878	0.8869
Geriatric_d	0.7572	0.9460	**0.9820**	0.8838	0.9175
Psychiatry_d	0.6946	0.9160	**0.9767**	0.7723	0.8998
Reception_d	0.6204	**0.8451**	0.7449	0.7184	0.7157
Stand-alone Tabu Search					
Instance	*MinWS*	*MinMax*	*MinDev*	*MinError*	*MinSS*
Emergency_i	0.8398	0.9821	**0.9900**	0.9274	0.9406
Geriatric_i	0.6949	0.9618	**0.9660**	0.7306	0.7320
Psychiatry_i	0.8693	0.9778	**0.9979**	0.9699	0.9721
Reception_i	0.7875	**0.8532**	0.8071	0.7995	0.7681
Emergency_d	0.7562	0.9711	**0.9921**	0.8784	0.9394
Geriatric_d	0.7256	0.9551	**0.9391**	0.8577	0.9245
Psychiatry_d	0.7101	0.8857	**0.9650**	0.7783	0.9048
Reception_d	0.5796	**0.8033**	0.7680	0.6397	0.7734

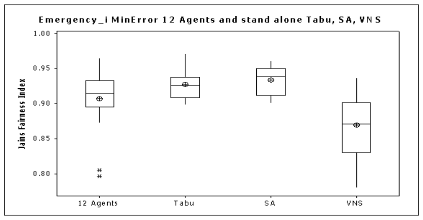
Table 4: The average Jains fairness index obtained over 20 runs for a given fairness-based objective function for the stand-alone case, where the bold entries indicate the best one for a given instance.



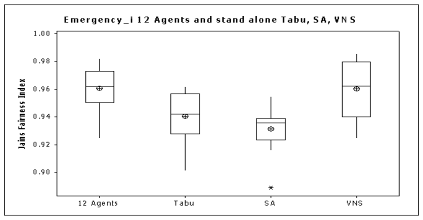
(a) MinMax



(b) MinDev



(c) MinError



(d) MinSS

Figure 3: The box plots of Scenario 1 algorithm and stand-alone metaheuristics based on Jains fairness index values obtained from 20 runs on Emergency_i for each fairness-based objective function.

We have re-evaluated all rosters produced at the end of each run by the Scenario 1 algorithm using a given fairness-based objective function with respect to the $MinWS$ objective function. For each instance, the average result obtained by Scenario 1 using $MinWS$ as objective function over 20 runs is used as a basis. Table 5 provides the average %gap (Equation 9) from the basis to the average $MinWS$ over 20 runs when a fairness-based objective function is used. The negative entries indicate improvement. Although MinDev and MinMax produce more fair rosters, the quality of these rosters is not as good in terms of the generic objective value, as expected (Smet et al., 2013b). In Scenario 1, it can be observed that $MinSS$ and $MinError$ performs better than the rest of the fairness-based objective functions on average, generating high quality roster with respect to $MinWS$. Scenario 1 using $MinError$ and $MinSS$ make 5-7% and 80-90% improvement upon the results of Scenario 1 algorithm using $MinWS$ for the Reception instances and Geriatric_i, respectively. On the other hand, $MinSS$ is one of the worst performing objective function in terms of fairness (Table 3).

Instance	$MinWS$	$MinMax$	$MinDev$	$MinError$	$MinSS$
Emergency_i	23245	367.55%	37.21%	22.88%	39.55%
Geriatric_i	3799	316.98%	427.11%	61.66%	85.34%
Psychiatry_i	29516	331.34%	315.43%	1.46%	8.45%
Reception_i	66719	39.36%	18.72%	-5.21%	-81.57%
Emergency_d	17477	465.85%	406.33%	39.54%	45.72%
Geriatric_d	38835	18.07%	74.55%	-6.99%	-90.03%
Psychiatry_d	11591	642.50%	600.85%	1.66%	41.61%
Reception_d	19018	71.84%	113.35%	-5.40%	-79.55%

Table 5: The average % gap from $MinWS$ to the $MinWS$ value of each fairness-based objective function for each instance.

5.2. Scenario 2 test results and comparison to Scenario 1

In the second test scenario, all the fairness models are used and tested at once. This is motivated by the idea that people planning a roster will have different concepts/models of fairness. If these models are allowed to cooperate, will the rosters be fair and obtained fast? Furthermore will this approach enable us to identify the best performing fairness-based objective function and guide search process? To this end the experiments are conducted in accordance with the first scenario in that 12 metaheuristic agents and 1 launcher employed under the cooperative search framework and 20 runs are performed for each instance. However, in this scenario three agents are grouped to form four different groups. Each group optimises with respect to one of the $MinMax$, $MinDev$, $MinError$ and $MinSS$ fairness-based objective functions, separately.

Table 6 provides the results for Scenario 2 and compares its performance to the best stand-alone metaheuristic, VNS using $MinMax$ and cooperating agents under Scenario 1 using $MinDev$ based on the average and best Jains fairness index over 20 runs. The Scenario 2 algorithm outperforms all stand-alone algorithms (Table 4) regardless of the objective function generating the most fair rosters for any given instance and this performance variation is statistically significant for all instances. Moreover, cooperative search under Scenario 2 consistently outperforms Scenario 1 using $MinDev$ or any other fairness based objective function on average and this performance difference is statistically significant for each instance. Scenario 1 and 2 produce the best rosters for three and four instances, respectively, with a tie in one instance. Figure 4 provides the box plots of Jains fairness index for the Geriatric_i instance over 20 runs considering cooperative and stand-alone metaheuristics using different objective functions as an example. VNS, Tabu and SA performs best using $MinDev$ on this instance. The results confirm the success of Scenario 2 over an approach with no cooperation even if it is used with the best performing objective function and this performance variation is statistically significant, except for Emergency_i. Still, Scenario 2 performs slightly better than Scenario 1 using $MinDev$.

	Best-of-runs			Average			
Instance	VNS	Scenario 1	Scenario 2	VNS	Scenario 1	vs.	Scenario 2
Emergency_i	0.9993	**0.9998**	0.9997	0.9966	0.9991	$\leq$	**0.9993**
Geriatric_i	0.9946	0.9978	**0.9980**	0.9646	0.9861	$<$	**0.9942**
Psychiatry_i	0.9997	**0.9999**	**0.9999**	0.9995	0.9995	$<$	**0.9997**
Reception_i	0.8439	0.9540	**0.9557**	0.8277	0.8515	$<$	**0.9528**
Emergency_d	0.9982	**0.9997**	0.9995	0.9955	0.9982	$<$	**0.9989**
Geriatric_d	0.9991	0.9993	**0.9998**	0.9981	0.9986	$<$	**0.9992**
Psychiatry_d	0.9994	**0.9999**	0.9997	0.9931	0.9988	$<$	**0.9992**
Reception_d	0.7727	0.9949	**0.9999**	0.7106	0.9824	$<$	**0.9986**

Table 6: Performance comparison of stand-alone VNS metaheuristic and scenario one using $MinDev$, and Scenario 2 based on the average and best-of-runs Jains fairness index values over 20 trials for each instance. Bold entries mark the best algorithm.

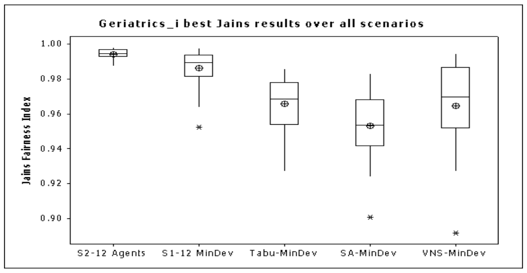


Figure 4: The box plot for Scenario 2 (S2) and Scenario 1 (S1), Tabu, SA and VNS using $MinDev$ based on Jains fairness index obtained from 20 runs for Geriatric_i.

Table 7 provides the average %gap (see Equation 9) from the basis (first column) in Table 5 to the average $MinWS$ over 20 runs under Scenario 2. This quantity indicates how $MinWS$ objective value changes while the fairness is optimised under Scenario 2. The results are consistent with the previous findings by Smet et al. (2013b), indicating that optimising $MinWS$ does not help much in terms of fairness. Scenario 2 is performing worse than Scenario 1 in the overall, regardless of the objective function used within the framework in terms of $MinWS$, yet it is much better at producing fair rosters as compared to Scenario 1.

Instance	$MinWS$	Scenario 2
Emergency_i	23245	340.60%
Geriatric_i	3799	501.93%
Psychiatry_i	29516	311.68%
Reception_i	66719	28.60%
Emergency_d	17477	456.62%
Geriatric_d	38835	67.49%
Psychiatry_d	11591	601.16%
Reception_d	19018	177.32%

Table 7: The average %gap over 20 run for a given objective function measured from the basis (first column of Table 5) to Scenario 2 for each instance.

6. Conclusion

This paper describes a flexible generic agent-based cooperative search framework to build fair nurse rosters. The cooperative search framework combines the strength of multiple metaheuristics and several fairness objective functions to generate high quality rosters. The agents improve the quality of local rosters using the same or different metaheuristic solution methods with the same or different fairness objective functions and parameter settings. The agents cooperate asynchronously using pattern matching and reinforcement learning in order to generate fair nurse rosters.

The experiments were conducted to evaluate the effectiveness of using cooperation to generate fair nurse rosters using existing real world benchmark problems. The experimental results demonstrated the success of cooperative search in producing fairer rosters. Furthermore, when all agents cooperate using the same fairness objective function, the function minimising the deviation from the average roster turned out to be superior, while the weighted sum objective function produced the least fair rosters. This observation confirms the results obtained by Smet et al. (2013b).

Finally, cooperative search with combined fairness models produced the fairest nurse rosters. Supposedly, this phenomenon can be ascribed to the fact that cooperation of the different fairness models introduced an element of efficiency. This could be in line with the strength of human cooperation, where the cooperation of nurses sharing different impressions of fair nurse rosters can lead to better rosters in terms of fairness. The results have confirmed that the cooperation of multiple decision makers who do not necessarily share the same models and the same heuristics produced fairer rosters than optimisation from a single planner's point of view. Future work will investigate the fairness issues in other problem domains.

References

G. R. 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*, 10(1):649–671, 2006.

F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi. Jadea java agent development framework. *Multi-Agent Programming*, pages 125–147, 2005.

D. Bertsimas and J. Tsitsiklis. Simulated annealing. *Statistical Science*, pages 10–15, 1993.

B. Bilgin, P. De Causmaecker, B. Rossie, and G. Vanden Berghe. Local search neighbourhoods to deal with a novel nurse rostering model. *Annals of Operations Research*, 194(1):33–57, 2012.

C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

A. L. Bouthillier and T. G. Crainic. A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Computers & Operations Research*, 32(7):1685–1708, 2005.

E. K. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe. Fitness evaluation for nurse scheduling problems. In *Proceedings of the Congress on Evolutionary Computation (CEC2001)*, pages 1139–1146, Seoul, Korea, May 27-30 2001. IEEE Press.

E. K. Burke, P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499, 2004.

C. C. B. Cavalcante, C. Carvalho de Souza, M. W. P. Savelsbergh, Y. Wang, and L. A. Wolsey. Scheduling projects with labor constraints. *Discrete Applied Mathematics*, 112(1):27–52, 2001.

S. H. Clearwater, T. Hogg, and B. A. Huberman. Cooperative problem solving. *Computation: The Micro and the Macro View*, pages 33–70, 1992.

T. Crainic and M. Toulouse. Explicit and emergent cooperation schemes for search algorithms. *Learning and intelligent optimization*, pages 95–109, 2008.

T. G. Crainic and M. Gendreau. Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics*, 8(6):601–627, 2002.

T. G. Crainic, M. Toulouse, and M. Gendreau. Synchronous tabu search parallelization strategies for multicommodity location-allocation with balancing requirements. *OR Spectrum*, 17(2):113–123, 1995.

T. G. Crainic, M. Toulouse, and M. Gendreau. Toward a taxonomy of parallel tabu search heuristics. *INFORMS Journal on Computing*, 9:61–72, 1997.

S. De Jong, K. Tuyls, and K. Verbeeck. Artificial agents learning human fairness. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 863–870. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

S. Even, A. Itai, and A. Shamir. On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM Journal on Computing*, 5(4):691–703, 1976.

F. Glover. Tabu search: a tutorial. *Interfaces*, pages 74–94, 1990.

T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

S. Haspeslagh, P. De Causmaecker, and G. Vanden Berghe. A multi-agent system handling personnel shortages in hospitals. In *Proceedings of the 4th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2009)*, MISTA, pages 693–695, Dublin, August 2009.

T. Hogg and C. P. Williams. Solving the really hard problems with cooperative search. In *Proceedings Of The National Conference On Artificial Intelligence*, pages 231–231, 1993.

R. Jain, Dah-Ming Chiu, and William R. Hawe. *A quantitative measure of fairness and discrimination for resource allocation in shared computer system.* Eastern Research Laboratory, Digital Equipment Corporation, 1984.

T. James, C. Rego, and F. Glover. A cooperative parallel tabu search algorithm for the quadratic assignment problem. *European Journal of Operational Research*, 195(3):810–826, 2009.

D. Julian, M. Chiang, D. O'Neill, and S. Boyd. Qos and fairness constrained convex optimization of resource allocation for wireless cellular and ad hoc networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 477–486. IEEE, 2002.

R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

S. Martin, D. Ouelhadj, P. Beullens, and E. Özcan. A generic agent-based framework for cooperative search using pattern matching and reinforcement learning. Technical report, University of Portsmouth, January 2012.

M. Muhlenthaler and R. Wanka. Fairness in academic course timetablling. In *Practice and Theory of Automated Timetabling (PATAT)*, pages 114–130, 2012.

D. Ouelhadj and S. Petrovic. A cooperative hyper-heuristic search framework. *Journal of Heuristics*, 16(6):835–857, 2010.

E. Özcan. Memetic algorithms for nurse rostering. In *Proceedings of the 20th international conference on Computer and Information Sciences*, ISCIS'05, pages 482–492. Springer-Verlag, 2005. ISBN 3-540-29414-7, 978-3-540-29414-6. doi: 10.1007/11569596_51. URL `http://dx.doi.org/10.1007/11569596_51`.

E Özcan. Memes, self-generation and nurse rostering. In EdmundK. Burke and Hana Rudov, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 85–104. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-77344-3. doi: 10.1007/978-3-540-77345-0_6. URL `http://dx.doi.org/10.1007/978-3-540-77345-0_6`.

S. Petrovic and G. Vanden Berghe. A comparison of two approaches to nurse rostering. *Annals of Operations Research*, 2012.

P. Smet, P. De Causmaecker, B. Bilgin, and G. Vanden Berghe. Nurse rostering: a complex example of personnel scheduling with perspectives. In *Automated Scheduling: Real World Case Studies*. Springer, 2013a.

P. Smet, S. Martin, D. Ouelhadj, E. Ozcan, and G. Vanden Berghe. Fairness in nurse rostering. Technical report, KU Leuven - KAHO Sint-Lieven and University of Portsmouth, 2013b.

M. Toulouse, K. Thulasiraman, and F. Glover. Multi-level cooperative search: A new paradigm for combinatorial optimization and an application to graph partitioning. *Euro-Par'99 Parallel Processing*, pages 533–542, 1999.

E. Vallada and R. Ruiz. Cooperative metaheuristics for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 193(2):365–376, 2009.

Sangsuree Vasupongayya and Su-Hui Chiang. On job fairness in non-preemptive parallel job scheduling. *Parallel and Distributed Computing and Systems (PDCS)*, (17), 2005.

Z.G. Wang and C. Wang. Automating nurse self-rostering: A multiagent systems model. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, volume 1–9 of *SMC 2009*, pages 4422–4425, 2009.