Noname manuscript No.
(will be inserted by the editor)

The Second International Nurse Rostering Competition

Sara Ceschia · Nguyen Dang · Patrick De Causmaecker · Stefaan Haspeslagh · Andrea Schaerf

the date of receipt and acceptance should be inserted later

Abstract This paper reports on the Second International Nurse Rostering Competition (INRC-II). Its contributions are (1) a new problem formulation which, differently from INRC-I, is a multi-stage procedure, (2) a competition environment that, as in INRC-I, will continue to serve as a growing testbed for search approaches to the INRC-II problem, and (3) final results of the competition. We discuss also the competition environment, which is an infrastructure including problem and instance definitions, testbeds, validation/simulation tools and rules. The hardness of the competition instances has been evaluated through the behaviour of our own solvers, and confirmed by the solvers of the participants. Finally, we discuss general issues about both nurse rostering problems and optimisation competitions in general.

Keywords Nurse rostering · Optimisation competition · Multi-stage optimisation

1 Introduction

Nurse rostering is a very important problem in healthcare management. Early papers date from the seventies, but especially in the last decade, it has attracted significant attention; see [4, 7] for a review of the literature and a classification.

The First International Nurse Rostering Competition (INRC-I) [12] was run in 2010. Since then, several research groups have taken this formulation and the

S. Ceschia · A. Schaerf
DPIA, University of Udine
via delle Scienze 206, I-33100, Udine, Italy
E-mail: {sara.ceschia,schaerf}@uniud.it

N. Dang · P. De Causmaecker
KU Leuven, Department of Computer Science, CODeS & imec-ITEC, KULAK
E. Sabbelaan 53, 8500 Kortrijk, Belgium
E-mail: {nguyenthithanh.dang, patrick.decausmaecker}@kuleuven.be

S. Haspeslagh
Vives University College, Commercial sciences and business management, MoBiz
Doorniksesteenweg 145, 8500 Kortrijk, Belgium
E-mail: stefaan.haspeslagh@vives.be

corresponding instances as a challenge [1,3,8,10,17,23,27] and produced remarkable results. Optimal solutions as well as new best solutions have also been found and reported.

The problem considered for INRC-I was the assignment of nurses to shifts in a fixed planning horizon, subject to a large number of hard and soft constraint types.

For the Second International Nurse Rostering Competition (INRC-II), we proposed a smaller set of constraint types, but within a *multi-stage* formulation of the problem. That is, the solvers of the participants were requested to deal with a sequence of cases, referring to consecutive weeks of a longer planning horizon (4 or 8 weeks).

The search method designed by the participants needs to be able to solve a single stage of the problem corresponding to one week. Some information, called *history*, is carried over between consecutive weeks, so that the one coming from the previous week has to be taken into account by the solver. The history includes *border* data, such as the last worked shift of each nurse, and *counters* for cumulative data, such as total worked night shifts. Counters' values have to be checked against global thresholds, but only at the end of the planning period. The planning horizon is not *rolling* [2] but fixed, in the sense that in the last week of the planning period all counters are compared to their limits and all discrepancies are added to the cost.

The problem formulation (constraints, objectives, and weights) has been based on the academic literature, with the aim of balancing simplicity and representativeness of real cases. The optimisation part is a simplified version of the formulation proposed in INRC-I [12], which incorporated more real-world constraints taken from the literature. In this competition, we keep the basic and important constraints that have been mentioned in several works, including [4,7,26], to name a few. The multi-stage part is added to reflect new developments in recent literature. Traditionally, a static planning horizon is solved without any consideration of past or future information. This is also the setting of INRC-I. However, in reality, the rostering of a current time horizon is usually influenced by the assignments of nurses in previous periods [11]. In [13,25], history information has been modeled and taken into account. On the other hand, optimising each single period separately might induce a bad overall roster when we look at the quality in the long-term. This issue was analysed in [22] and a setting called *stepping horizon*, in which the evaluation is done over multiple horizons, was proposed. The multi-stage approach in INRC-II is similar to this setting.

We provided a simple command-line simulation/validation software to be used to simulate the solution process and to evaluate the quality of the solver. The *simulator* invokes the participant's solver for each stage iteratively, then updates the history after each single execution. The provided *validator* concatenates the solutions for all weeks, and evaluates them all together, along with the cumulative data coming from the *final history*, which is the history produced by the schedule of the last week.

The solver needs to take into account the following separate input sources:

Scenario: Information that is global to all weeks of the entire planning horizon, such as nurse contracts and shift types.

Week-data: Specific data of the single week, like daily coverage requirements and nurse preferences for specific days.
History: Information that must be passed from one week to another, in order to compute constraint violations properly. It includes border information and global counters.

The solver must deliver an output file, based on which the simulator computes the new history file to be passed back to the solver for the solution of the next week. As will be explained further, besides the mandatory input and output files, a custom data file may be used to exchange information between two stages.

The paper is organised as follows. Section 2 defines the problem. Section 3 describes the testbeds. Section 4 shows the software tools made available to the participants to evaluate their solver and Section 5 discusses the rules of the competition. Section 6 shows the results of both the preliminary round and the final round. Finally, a discussion and conclusions are given in Section 7.

All information about the competition, including a mathematical formulation for the proposed multi-stage nurse rostering problem, is available at `http://mobiz.vives.be/inrc2/`.

2 Problem definition

The basic (one-stage) problem consists in the weekly scheduling of a fixed number of nurses using a set of shifts, such that in each day a nurse works a shift or has a day off. Nurses may have multiple skills, and for each skill we are given different coverage requirements.

Given the multi-stage nature of the overall process, the input data of the problem comes from three different sources, called scenario, week data, and history, as explained in the following sections.

2.1 Scenario

The scenario represents the general data common to all stages of the overall process. It contains the following information:

Planning horizon: The number of weeks that compose the planning period.
Skills: The list of skills included in the problem (head nurse, regular nurse, trainee, ...). Each nurse has one or more skills, but in each working shift she/he covers only one skill request.
Contracts: Each nurse has one specific contract (full time, part time, on call, ...). The contract sets limits on the distribution and the number of assignments within the planning horizon. In detail, it contains:
- minimum and maximum total number of assignments in the planning horizon;
- minimum and maximum number of consecutive working days;
- minimum and maximum number of consecutive days off;
- maximum number of working weekends in the planning horizon;
- a Boolean value representing the presence of the *Complete weekend* constraint to the nurse, which states that the nurse should work both days of the weekend or neither of them.

Nurses: For each nurse, the name (identifier), the contract and the set of skills are given.

Shift types: For each shift type (early, late, night, ...), the minimum and maximum number of consecutive assignments of that specific type, and a matrix of forbidden shift type successions is given. For example, it may not be permissible to assign a nurse an early shift the day after a late one.

2.2 Week-data

The week-data contains the specific data for the single week. It consists of the following information:

Requirements: for each shift, for each skill, for each week day, the optimal and minimum number of nurses necessary to fulfil the working duties.

Nurse requests: a set of triples, each one is composed of the nurse's name, the week day, and a shift. The presence of a given triple represents the request of the nurse not to work on a given shift on a given day. The special shift name `Any` represents the request of not to work on any shift that day, i.e. have a day off.

The above information varies from week to week, due to variability in the number of current patients and the specific preferences of nurses.

Conventionally, all weeks start with Monday, so that the data is stored in the order `Mon`, `Tue`, ..., `Sun`.

Note that requests, as explained in Section 2.5, are treated as *soft* constraints, and can be rejected. Requirements instead are both hard (the minimum) and soft (the optimal number) constraints.

2.3 History

The history contains the information that must be carried over from one week to the following one, so as to evaluate the constraints correctly. In detail, it reports the following two types of information for each nurse:

Border data: The border data is used to check the constraints on consecutive assignments. They are:
- shift worked in the last day of the previous week, or the special value `None` if the nurse had a day off;
- number of consecutive worked shifts of the same type and number of consecutive worked shifts in general (both are 0 if the last worked shift is `None`);
- number of consecutive days off (0 if the last worked shift is not `None`).

Counters: The counters calculate the cumulative values over the weeks that have to be compared with their limits, only at the end of the planning period. They are:
- total number of worked shifts;
- total number of worked weekends.

The history is computed after each week based on the solution delivered by the solver and on the previous history. The history passed to the solver for the first week has all counter values equal to 0, whereas the border data can have any value.

2.4 Solution

The full solution process is a loop that is executed at each step by the solver for a week, repeating for all weeks in the planning period (4 or 8 weeks).

After each week, the history information is computed based on the solution and the previous history, and delivered in a new file. This is done by the simulator, as the solver provides only the solution itself.

The complete process is sketched in Figure 1(a), assuming a scenario of 4 weeks. Input files are coloured in different shades of cyan, and the output ones in different shades on magenta.

In addition to generating a solution file after solving each stage, the solver might want to save some other prediction information in a custom file (in yellow in Figure 1) and pass it to the next solver call, in order to guide the solving of the next stage. The content and the format of this file is free, but its name is set by the simulator described in Section 4.1.

The output produced by the solver is a list of assignments of nurses to shifts and skills. Each entry contains the nurse name, the week day, the shift, and the skill. As an example, consider the entry ⟨`Mary`, `Tue`, `Night`, `HeadNurse`⟩, that states that the nurse `Mary` works on Tuesday on the night shift in the role of head nurse.

The quality of the overall solution is evaluated, as shown in Figure 1(b), for the entire planning horizon based on:

- the solution for each week containing the assignments of the nurses to shifts and skills, using the requirements in the week-data file and the border data in the history file;
- the counters of the final history file, against the limits provided in the scenario.

2.5 Constraints

According to the setting outlined above, we split the constraints into two sets: those that can be computed for each week separately, and those that are computed only globally at the end of the planning period.

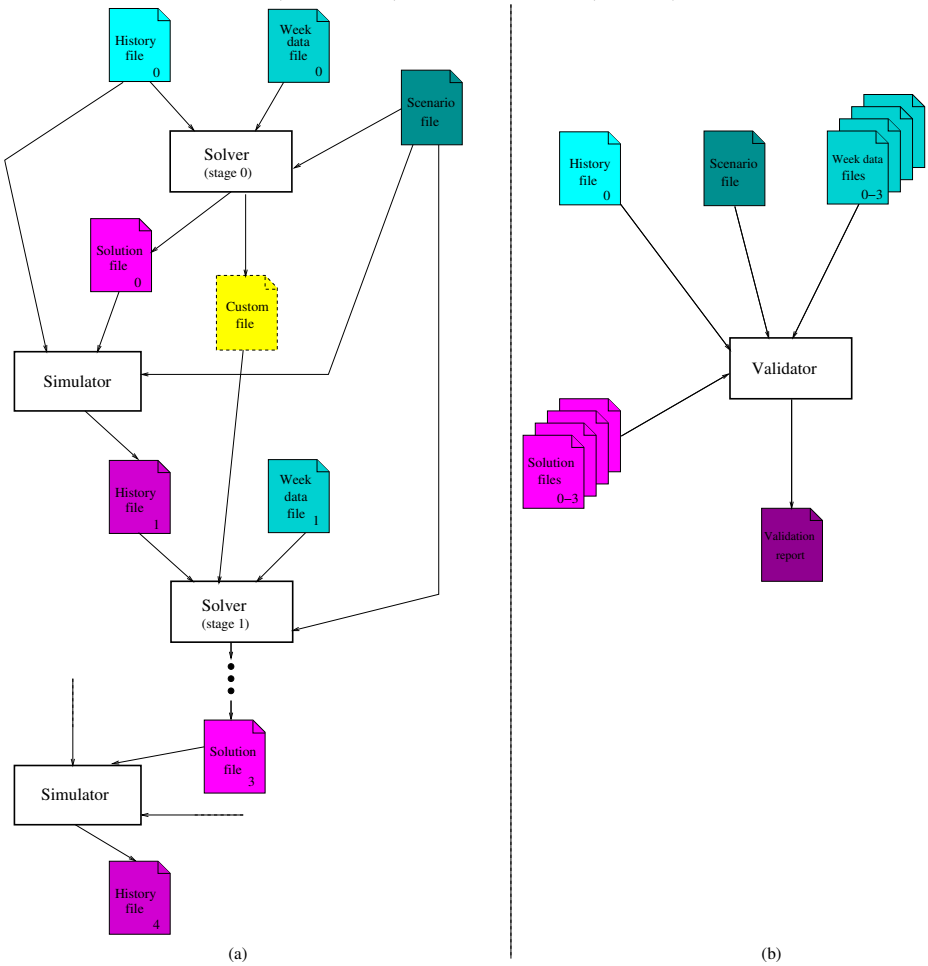
As customary, they are also split into hard and soft constraints. The former must be always satisfied, while the latter contribute to the objective function.

Constraints, objectives, and weights have been designed based on the academic literature, with the aim of balancing simplicity and representativeness of real cases.

2.5.1 Constraints on the single week

Below is the list of hard (H) and soft (S) constraint types. The weight of each single soft constraint is shown prior to its description.

Fig. 1 The overall solution/simulation/validation process (4 weeks).



H1. Single assignment per day: A nurse can be assigned to at most one shift per day.
H2. Under-staffing: The number of nurses for each shift for each skill must be at least equal to the minimum requirement.
H3. Shift type successions: The shift type assignments of one nurse on two consecutive days must not violate the forbidden successions as provided in the scenario.
H4. Missing required skill: The shift of a given skill must necessarily be completed by a nurse having that skill.
S1. Insufficient staffing for optimal coverage (30): The number of nurses for each shift for each skill must be equal to the optimal requirement. Each missing nurse is penalised according to the weight provided. Extra nurses above the optimal value are not considered in the cost.

S2. Consecutive assignments (15/30): A minimum and maximum number of consecutive assignments to the same shift type should be respected. Each extra or missing day is multiplied by the corresponding weight. The weight for consecutive shift of the same type is 15.
Similarly, a minimum and maximum number of consecutive assignments to any working shift should be respected. For consecutive working days of any shift the weight is 30. In both cases, the evaluation involves also the border data.
S3. Consecutive days off (30): Minimum and maximum number of consecutive days off should be respected. Their evaluation involves also the border data. Each extra or missing day is multiplied by the corresponding weight.
S4. Preferences (10): Each assignment to an undesired shift is penalised by the corresponding weight.
S5. Complete weekend (30): Every nurse that has the complete weekend value set to *true*, must work both weekend days or neither. If she/he works only one of the two days `Sat` and `Sun` this is penalised by the corresponding weight.

2.5.2 Constraints spanning the planning horizon

The following (soft) constraints are evaluated only at the end of the planning period:

S6. Total assignments (20): For each nurse the total number of assignments (working days) must be included within the limits (minimum and maximum) enforced by her/his contract. The difference (in either direction), multiplied by its weight, is added to the objective function.
S7. Total working weekends (30): For each nurse the number of working weekends must be less than or equal to the maximum. The number of worked weekends in excess is added to the objective function multiplied by the weight. A weekend is considered "working" if at least one of the two days (`Sat` and `Sun`) is busy for the nurse.

Obviously, the solver should take constraints S6 and S7 into account in each single stage. However, their violation values have a decreasing degree of uncertainty going from one week to the following one, and only in the last week can they be evaluated exactly. It is up to the solver to decide the way to model them in the cost function in the different weeks, possibly making forecasts on the forthcoming ones.

3 Public and hidden testbeds

The complete solution process requires as input a scenario, an initial history, and data for four or eight weeks, and it produces solutions for four or eight weeks, together with one final history. Scenario, week-data, history, and week-solutions are written in separate files each one with its own syntax.

For ease of processing, all files are provided in XML, JSON, and text-only formats, and each participant could use the format that (s)he considered most convenient for her/his implementation. File formats are explained in [5].

3.1 Datasets

Files belonging to the same case are grouped in a *dataset*, which is composed of the following set of files:

- 1 scenario file;
- 3 initial history files;
- 10 week-data files.

The participants of the competition were provided with a *public* testbed composed of 14 datasets, one for each combination of number of weeks and number of nurses, taken from the sets {30,40,50,60,80,100,120} and {4,8}, respectively. Datasets are named using these two numbers with the prefixes `n` (for nurses) and `w` (for weeks). For example, the dataset `n050w8` is the (unique) one with 50 nurses and 8 weeks.

In addition, three test datasets, `n005w4`, `n012w8`, and `n021w4`, were provided for testing and debugging purposes.

A different *hidden* testbed has been used for the final stage of the competition. This is composed of 6 datasets having the number of nurses and weeks in the sets {35,70,110} and {4,8}. This testbed was not shown to the participants until the end of the competition.

Both testbeds are made of artificial data, but they were created starting from realistic patterns and distributions. These patterns and distributions link the hidden testbed to the public one.

We created the scenario files based on expert knowledge, trying to make them close to realistic cases in hospitals of different sizes. The week data files were automatically generated based on the information provided by the corresponding scenario. The initial history files were obtained using the solvers we had developed for the proposed multi-stage problem before we launched the competition. This is described in more detailed in Section 3.3.

3.2 Instances

An *instance* is a combination of a specific scenario, an initial history, and a sequence of 4-/8-week-data files, all belonging to the same dataset. The same week-data file can be used many times in the same instance.

For example, the string `n040w4_2_6-1-0-6` identifies the instance belonging to the dataset `n040w4`, completed with the history file number `2` and the sequence of week-data files number `6`, `1`, `0`, and `6`.

Given that any week-data file can be used in any week of the planning horizon, for each dataset of w weeks, the number of distinct instances that can be created is $10^w \times 3$.

From each dataset of the public testbed, 2 instances were randomly selected for the competition (28 instances altogether) and given to the participants two weeks before the deadline for the submission of the results. Similarly, from each dataset of the hidden testbed, 10 instances were randomly selected for the final stage of the competition (60 instances altogether).

For the test datasets we randomly selected one instance each, for which we provided also a solution that could be used for inspection and comparison.

3.3 Solving approaches and data generation procedure

We have developed two solvers for testing our instances. The first one is an exact method based on the problem formulation proposed in [23] and makes use of the MILP solver CPLEX (v.12.5) [6]. The second one uses a Simulated Annealing (SA) technique, with a neighbourhood relation based on the union of swap and replace moves. This one is implemented using EasyLocal++ (v.3) [9].

Both MIP and SA algorithms were used as *dynamic* solvers, i.e., solving week by week as required by the multi-stage setting of the competition. We devised some simple strategies for dealing with the two global objectives on cumulative data (the minimum and maximum total number of assignments and the maximum number of working weekends for each nurse):

- For the MIP solver: the bound values are equally divided among the weeks for the constraint on the total number of assignments, and are kept as original for the constraint on the total number of working weekends.
- For the SA solver: at each stage, for every nurse we estimate their current bounds on the number of assignments based on their history, their contract and the number of remaining weeks in the planning horizon. The computed limits are rational values that are then suitably rounded according to parametric thresholds.

The SA algorithm is used also for the *static* problem, i.e., the whole instance was solved at once with all the week data available. The percentage gap between solutions produced by the dynamic solvers and the static one can be considered as an experimental indicator for the importance of the lack of information in the multi-stage setting. On all instances tested, we consistently observed a relatively large gap. In addition, results on all tested instances confirmed that:

- all instances are feasible, i.e., there is at least one solution that satisfies all hard constraints;
- the solutions produced by the SA solver (which is stochastic) have a relatively large variance; this behaviour suggests that the instances are quite challenging and not straightforward.

Another usage of the solvers was to generate the initial history files of the competition's testbeds, by running the solver for a few weeks and collecting the final history. Compared to "zero" initial history files, in which all nurses were assumed not to have worked at all during the previous planning period and their values of consecutive non-working shifts and days at the border are set to the corresponding lower bounds, these initial history files were not only more realistic, but also induced a large increase in the percentage gap between the dynamic and the static solvers.

4 Tools

We provided the participants with a suite of software tools. The simulator manages the multi-stage solution process. The validator certifies the quality of a given instance. The benchmark executable computes the allowed running time for each

computer. Furthermore, the feasibility checker gives the opportunity to the participant to check whether or not a specific instance of a given dataset has a feasible solution.

4.1 Simulator and validator

As shown in Figure 1, the simulator receives a scenario file, an initial history file, the solver's executable file name and a sequence of week-data files as its input. It then applies the solver on each week-data file, generating a history file for each stage based on the solution obtained from each solver call.

Besides the basic input described above, the user can also specify random seeds for his/her solver, the directory where the solver is run, and the directory where all solution files, generated history files, log files (the solver's console output after each solver call) and the validator's results are saved. In addition to generating a solution file after solving each stage, the solver might want to save some other information in a custom file and pass it to the next solver call, in order to guide the solving of the next stage better. This possibility is supported by the simulator.

The validator checks for the validity of a solution of an instance, and calculates the corresponding objective function value, according to the constraints' weights described in Section 2.4. The validator is automatically called by the simulator at the end of the solving procedure.

The specific command-line parameters of the simulator and the validator are explained in [5].

4.2 Benchmark

The benchmark program was designed to test how fast a machine is at doing the sort of things that are involved in rostering. For each problem size, which is defined as the number of nurses in the scenario file, the program states how long a participant could run the algorithm for each stage.

The benchmark is only suitable for individual, single processor machines. It is not suitable, for example, for specialist parallel machines or clusters. In general, for multi-core machines, one single core is allowed for the competition.

The program reports how long it took, and hence the time one can run her/his rostering algorithm per stage, for each number of nurses. On a relatively modern PC, the benchmark program will grant the participant approximately $10+3*(N-20)$ seconds for each stage, in which N is the number of nurses.

4.3 Feasibility checker

It could be the case that some instance created from a given dataset is infeasible. In order to prevent participants from waste time searching for feasible solutions when they do not exist, a feasibility checker was provided. The feasibility checker simply verified whether hard constraints are satisfiable in all weeks on the planning horizon.

5 Competition rules, dates, and adjudication procedure

The full set of rules is listed in [5]. We discuss here some of the most important ones. First, the use of third-party software is allowed only if it is free software and its behaviour is documented (Rule 4). This rule excludes the use of commercial software, like CPLEX and Gurobi; although they are available free of charge for universities, they are not free for all users.

Rule 5 states that the solver should run on a single core of the machine, so that the use of parallel solvers is forbidden. The organisers were aware that this limitation excludes algorithms exploiting the benefits of parallel investigation of different possibilities. The main reason for this rule is that it would have been difficult to reproduce and compare software using different multi-core architectures and environments.

Another important point (Rules 10 and 12) is that the adjudication is a two-step procedure. First, the set of finalists is selected based on results provided on the public testbed. Second, the software of all finalists is run on organisers' PCs and the final ranking among the finalist is based only on their results on the hidden instances. This procedure is meant mainly to reduce the burden of running all solvers, which could have been quite onerous in terms of both computational power and human effort (installing, compiling, ...).

The competition started on October 17, 2014. On that date, we released the public testbed, the specification paper [5], and the software tools. On May 15, 2015, the list of specific instances of the public testbed were released. The deadline for submission of participants' best results and their solvers was June 1, 2015. Notifications of the finalists was sent out on July 15, 2015. The winners were announced at the MISTA 2015 Conference in Prague (August 25-27, 2015).

The competition followed the same adjudication procedure of INRC-I [12], which in turn was imported from the Second International Timetabling Competition (ITC-2007) [18], and also used in the Third International Timetabling Competition (ITC-2011) [19]. The full procedure, which is based on ranks rather than actual scores, is explained in [5].

According to the rules, the number of finalists should have been set at 5. However, given the small difference between the 5th and two followers, they were augmented to 7.

6 Results

The competition experienced 105 registered accounts (necessary to access the data) and 27 google group members. The number of teams that went all the way to the submission stage was 15, coming from Europe, North and South America, and Asia.

All 15 participants supplied their solutions for the 28 instances extracted from the public datasets. As already mentioned, the specific instances had been made available to the participants 2 weeks before the deadline for submission.

The organisers selected the 7 finalists, out of the 15 participants, based on the ranks of the submitted results on these 28 instances. The finalists, along with the

Team	Participants	Institution (Country)	Search method
Hust.Smart	Zhouxing Su Zhuo Wang Zhipeng Lü	Huazhong University (CN)	Tabu Search [28]
LabGOL	Federica Picca Nicolino Francesco Bagattini Luca Bravi Niccolò Bulgarini Alessandro Galligari Fabio Schoen	Università di Firenze (IT)	Metaheuristics
NurseOptimizers	Michael Römer Taieb Mellouli	Martin Luther University Halle-Wittenberg (DE)	Network flow (MILP) [20]
ORTEC	Hujin Jin Gerhard Post Egbert van der Veen	ORTEC (NL)	Metaheuristics [14]
Polytechnique Montreal	Antoine Legrain Jérémy Omer Samuel Rosat	Polytechnique Montreal (CA)	Column generation [16] (MILP)
SSHH	Ahmed Kheiri	University of Exeter (UK)	Hyper-heuristic [15]
ThreeJohns	Ioannis X. Tassopoulos Ioannis P. Solos Grigorios N. Beligiannis	University of Patras (GR)	Variable neighbourhood search

Table 1 Finalists

Position	Team	Rank sum
1	NurseOptimizers	1.76
2	Polytechnique Montreal	1.86
3	SSHH	2.84
4	Hust.Smart	3.75
5	ORTEC	5.35
6	LabGOL	6.13
7	ThreeJohns	6.32

Table 2 Final outcome

search methods they used are listed (alphabetically) in Table 1. A brief description of the solution approaches adopted by each finalist can be found in Paragraph 6.1.

For the hidden testbed, each solver was run on organisers' PCs 10 times for each of the 60 instances (10 per dataset), and all results were grouped for the final ranking.

The final outcome is the one in Table 2, which shows also the average rank obtained by each solver. A spreadsheet showing the details of each run on each instance is available for download from the competition website.

Figure 2 shows box-plots of the normalised cost of the finalists on instances belonging to each hidden dataset. The winner, team NurseOptimizers, delivered many infeasible solutions on the dataset of `35` nurses and `8` weeks due to some instability of the solver. However, this does not affect their overall ranking a much since the rank-based evaluation does not penalise severely infeasible solutions (of course, the team was still ranked the last on those instances). In general, the rankings are quite consistent among all datasets, especially between the top four and the rest. We also observed similar consistency on the late instances.

Fig. 2 Normalised cost of the finalists on the hidden instances. Each plot is for a combination of a number of nurses and a planning horizon size. Box-plots in each plot are shorted according to their corresponding team's average ranking over the whole hidden instance set.

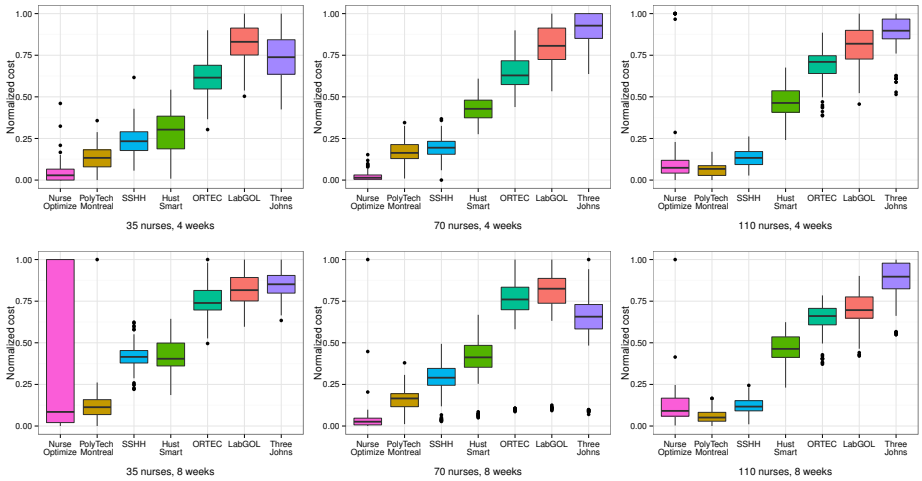


Table 2 shows that the best results were obtained by IP-based methods. In fact, the first two in the ranking used MILP techniques, resulting in quite a large advantage compared to the solvers using metaheuristics.

This is quite a surprise, given that previous competitions on timetabling and nurse rostering had always been won by means of metaheuristics.

In our opinion, an important role is played by the constraints on consecutiveness (S2 and S3), which make the problem hard in general. The relatively short length (7 days) of each stage made the use of IP-based techniques far more practicable, even in the presence of S2 and S3 type constraints, whereas in the longer stages, metaheuristics might have gained the advantage.

An important role is also played by the forecasting strategy. Looking at the solutions proposed, we see that there are two main directions for dealing with the multi-stage nature of the problem: the first one is by adding auxiliary constraints into the current formulation [14], and the second one is to extend the rostering period at each stage and solve the current stage together with the anticipated stages [15, 20]. The second one seems to work better in the competition context, since the top three all exploited ideas in this direction.

6.1 Finalists' search methods

We provide here a brief description of the search methods used by the finalists. We refer to the specific papers by the authors themselves for the detailed description.

– The Hust.Smart [28] group proposed an iterated Tabu Search approach that, starting from a greedy initial solution, alternates intensification and diversification phases.

- LabGOL used a combination of different metaheuristics (Tabu Search, pure Local Search, Variable Neighbourhood Search, ...), integrated in a hyper-heuristic framework which implements Iterated Local Search and Multi Start techniques.
- NurseOptimizers [20,21] reformulated the problem as a network flow-based MILP. The flow model associates each nurse with a directed acyclic graph in which arcs correspond to shift and days off assignments; a flow from the source to the sink can be interpreted as a roster. The solver employs a deterministic look-ahead approach, that relaxes the integrality constraint, and extends the planning period with an anticipation period, whose data is artificially generated based on the demand information of the current and the previous weeks.
- The solver presented by the ORTEC group [14] is an adaptation of the commercial solution ORTEC Workforce Scheduling, which is based on local search techniques. In detail, they added some artificial soft constraints that ensure a feasible connection between two consecutive weeks.
- The Polytechnique Montreal [16] team modeled the problem as a mixed integer program, and solved it by a column generation procedure. To anticipate future requirements, the costs are modified and constraints are added; then, at each stage, several possible schedules are generated and evaluated, and the best one is selected.
- SSHH [15] proposed a Sequence-based Selection Hyper-Heuristic that utilises a hidden Markov model; each of the nine low level heuristics has a transition probability matrix to move from it to another low level heuristic and the emission probability matrix to select the heuristic parameter and the acceptance strategy. In addition, instead of solving the current stage solely, the algorithm generates the demand of the coming weeks based on current information and solve the whole anticipated horizon.
- The search method implemented by the ThreeJohns group uses a modified variable neighbourhood search, with an ad-hoc procedure to escape from local optima.

7 Discussion, future work, and conclusions

In this paper, we have described the progress and the outcome of the second International Nurse Rostering Competition (INRC-II), which is the first one in the field of rostering and timetabling that involves a certain degree of uncertainty in the objective function.

In our opinion, given the originality and the complexity of the competition ground, the presence of 15 submissions can be considered as a good success. In addition, the quality of the solutions offered by the participants is very high and the results obtained will be quite hard to outperformed in the future.

We believe that competitions are very useful to the optimization community in order to stimulate the design of many search techniques on a common ground, which could lead to comparison and cross-fertilization of different paradigms. The overall aim of a competition is not only to highlight the most promising techniques, but also to lead toward the development of new hybrid ones.

Indeed, many competitions have been proposed to the optimization field. The most famous and seminal ones are the DIMACS challenges (`dimacs.rutgers.edu/`

Challenges/) that started back in 1990. DIMACS challenges have been active until 2014 and have dealt with a large variety of optimization problems. Also long-running and influential are the SAT-competition series (www.satcompetition.org), which started in 2002 and focused on the SAT problem, but indirectly also on many other problems that have been coded into SAT. More recently, we have observed the raise of many other series of competitions; among other, we have Verolog (www.verolog.eu) on vehicle routing, ROADEF (challenge.roadef.org) on a set on complex, real-world problems, BBComp (bbcomp.ini.rub.de) that focuses on the black-box aspect of optimization, and the new OASC competition on algorithm selection (www.coseal.net/open-algorithm-selection-challenge-2017-oasc/). As mentioned before, the closest to Nurse Rostering competitions, in terms of problem domain, is the Timetabling Competition series [18,19], that started in 2002 and regards educational timetabling problems. Outside the optimization boundaries, we could also mention the classical TREC competitions (trec.nist.gov/pubs.html), starting in 1992 and still enduring, on information retrieval.

For the future, we plan to propose several extensions and variants of the problem with the aim of capturing many more real-world situations. The first extension that we have already pursued is the case with multi-week stages. That is, the single stage could involve not only one single week, but a longer time-frame consisting of 2 or 4 weeks. Accordingly, we have added to the competition website new datasets with an overall planning horizon of 16 and 32 weeks, that could be used as test cases for longer stages.

In addition, we plan to consider the alternative rolling horizon formulation, in which long-term objectives are not checked at prefixed times, but monitored continuously.

To conclude, we would like to provide some comments on optimisation competitions in general, based on our experience on the organisation of INRC-II.

First of all, we observed that the initial ranking, based on solutions provided by the participants, was substantially preserved in the final round. This shows that participants have neither *overtuned* their solvers on the public testbed nor took advantage of the so-called *Mongolian horde* approach [24] ("Run as many trials as you can and report only the best of all of them").

Secondly, we realised, based on comments from the participants, that some rules are quite debatable. The most controversial one is, in our opinion, the prohibition of multi-core architectures. As already mentioned, the main reason was the technical difficulty of reproducing the execution. This could be addressed in future competitions by using standardised virtual machines provided to all participants by the organisers.

The other questionable point is the possibility of using commercial tools, such as CPLEX or Gurobi, which are quite popular but currently available for free only to universities. The main motivation for this decision is to promote the dissemination of free software.

Obviously, we believe that approaches using commercial software will be most welcome in the research community; in fact, one of our own solvers uses CPLEX, but we preferred to leave them out of the competition frame. An alternative, more stringent criterion could have been the restriction to open-source software only, so that the detailed behaviour of the algorithms would have been available for analysis.

References

1. Mohammed A. Awadallah, Asaju La'aro Bolaji, and Mohammed Azmi Al-Betar. A hybrid artificial bee colony for a nurse rostering problem. *Applied Soft Computing*, 35:726–739, 2015.
2. Jonathan F Bard and Hadi W Purnomo. Short-term nurse scheduling in response to daily fluctuations in supply and demand. *Health Care Management Science*, 8(4):315–324, 2005.
3. Edmund K Burke and Tim Curtois. New approaches to nurse rostering benchmark instances. *European Journal of Operational Research*, 237(1):71–81, 2014.
4. Edmund K Burke, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499, 2004.
5. Sara Ceschia, Nguyen Dang Thi Thanh, Patrick De Causmaecker, Stefaan Haspeslagh, and Andrea Schaerf. Second international nurse rostering competition (INRC-II), problem description and rules. *CoRR*, abs/1501.04177, 2015.
6. CPLEX. CPLEX Optimizer. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`, 2012. v. 12.5.
7. Patrick De Causmaecker and Greet Vanden Berghe. A categorisation of nurse rostering problems. *Journal of Scheduling*, 14(1):3–16, 2011.
8. Federico Della Croce and Fabio Salassa. A variable neighborhood search based matheuristic for nurse rostering problems. *Annals of Operations Research*, 218(1):185–199, 2014.
9. Luca Di Gaspero and Andrea Schaerf. EasyLocal++: An object-oriented framework for flexible design of local search algorithms. *Software—Practice and Experience*, 33(8):733–765, 2003.
10. Martin Josef Geiger. Personnel rostering by means of variable neighborhood search. In *Operations Research Proceedings 2010*, pages 219–224. Springer, 2011.
11. Celia A Glass and Roger A Knight. The nurse rostering problem: A critical appraisal of the problem structure. *European Journal of Operational Research*, 202(2):379–389, 2010.
12. Stefaan Haspeslagh, Patrick De Causmaecker, Andrea Schaerf, and Martin Stølevik. The first international nurse rostering competition 2010. *Annals of Operations Research*, 218:221–236, 2014.
13. Atsuko Ikegami and Akira Niwa. A subproblem-centric model and approach to the nurse scheduling problem. *Mathematical programming*, 97(3):517–541, 2003.
14. Hujin Jin, Gerhard Post, and Egbert van der Veen. ORTEC's contribution to the second international nurse rostering competition. In *Proc. of the 11th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2016)*, pages 599–501, 2016.
15. Ahmed Kheiri, Ender Özcan, Rhyd Lewis, and Jonathan Thompson. A sequence-based selection hyper-heuristic: Case study in multi-stage nurse rostering problem. In *Proc. of the 11th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2016)*, pages 503–505, 2016.
16. Antoine Legrain, Jérémy Omer, and Samuel Rosat. A rotation-based branch-and-price approach for the nurse scheduling problem. Working paper, available at `https://hal.archives-ouvertes.fr/hal-01545421`, 2017.
17. Zhipeng Lü and Jin-Kao Hao. Adaptive neighborhood search for nurse rostering. *European Journal of Operational Research*, 218(3):865–876, 2012.
18. Barry McCollum, Andrea Schaerf, Ben Paechter, Paul McMullan, Rhyd Lewis, Andrew J. Parkes, Luca Di Gaspero, Rong Qu, and Edmund K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22(1):120–130, 2010.
19. Gerhard Post, Luca Di Gaspero, Jeffrey H. Kingston, Barry McCollum, and Andrea Schaerf. The third international timetabling competition. *Annals of Operations Research*, 239(1):69–75, 2016.
20. Michael Römer and Taïeb Mellouli. A direct MILP approach based on state-expanded network flows and anticipation for multi-stage nurse rostering under uncertainty. In *Proc. of the 11th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2016)*, pages 549–551, 2016.
21. Michael Römer and Taïeb Mellouli. Future demand uncertainty in personnel scheduling: Investigating deterministic lookahead policies using optimization and simulation. In *Proceedings of the 30th European Conference on Modelling and Simulation (ECMS 2016)*, pages 502–507, 2016.

22. Fabio Salassa and Greet Vanden Berghe. A stepping horizon view on nurse rostering. In *Proc. of the 9th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2012)*, pages 161–173, 2012.
23. Haroldo G. Santos, Túlio A. M. Toffolo, Rafael A. M. Gomes, and Sabir Ribas. Integer programming techniques for the nurse rostering problem. *Annals of Operations Research*, 239(1):225–251, 2016.
24. Andrea Schaerf and Luca Di Gaspero. Measurability and reproducibility in timetabling research: Discussion and proposals. In E. Burke and H. Rudová, editors, *Proc. of the 6th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2006), selected papers*, volume 3867 of *Lecture Notes in Computer Science*, pages 40–49, Berlin-Heidelberg, 2007. Springer-Verlag.
25. Pieter Smet, Burak Bilgin, Patrick De Causmaecker, and Greet Vanden Berghe. Modelling and evaluation issues in nurse rostering. *Annals of Operations Research*, 218(1):303–326, 2014.
26. Pieter Smet, Peter Brucker, Patrick De Causmaecker, and Greet Vanden Berghe. Polynomially solvable personnel rostering problems. *European Journal of Operational Research*, 249(1):67–75, 2016.
27. Ioannis P Solos, Ioannis X Tassopoulos, and Grigorios N Beligiannis. A generic two-phase stochastic variable neighborhood approach for effectively solving the nurse rostering problem. *Algorithms*, 6(2):278–308, 2013.
28. Su Zhouxing, Wang Zhuo, and Lü Zhipeng. Weighted tabu search for multi-stage nurse rostering problem. *SCIENTIA SINICA Informationis*, 46(7):834–854, 2016.