New Benchmark Instances for the Staff Rostering Problem

Kimmo Nurmi, Jari Kyngäs and Nico Kyngäs

Abstract— **Workforce management has become increasingly important for companies. It is vital to optimize the performance of staff on both financial efficiency and customer satisfaction. Furthermore, it is important to optimize employer and employee satisfaction simultaneously. Good staff schedules are very important for the welfare of the staff, resulting in increased job satisfaction and reduction of sick-leaves. Good optimization methods are needed to roster the staff efficiently. This paper defines the core staff rostering problem, introduces the necessary terminology and details the constraints of the problem. The first set of random benchmark instances for the problem is introduced. The instances are simplified to the point where the used constraints comprise the core of almost any staff rostering problem. This enables a large number of researchers to find out how well their optimization methods work. We publish the best solutions we have found. The instances are available online.**

***Index Terms*—Staff rostering, scheduling, benchmark instances, PEAST algorithm**

I. Introduction

Workforce scheduling, also called staff scheduling and labor scheduling, is a difficult and time consuming problem that every company or institution that has employees working on shifts or on irregular working days must solve. Workforce optimization is the key to efficient use of workforce, customer satisfaction, employee satisfaction and control over rapidly changing situations, duties and competence demands. The workforce scheduling problem has a fairly broad definition. Most of the studies focus on assigning employees to shifts, determining working days and rest days or constructing flexible shifts and their starting times. Different variations of the problem and subproblems are NP-hard and NP-complete [1]-[6], and thus extremely hard to solve. The first mathematical formulation of the problem based on a generalized set covering model was proposed by Dantzig [7]. Good overviews of workforce scheduling are published by Alfares [8], Ernst et al. [9], Meisels and Schaerf [10] and De Causmaecker and G. Vanden Berghe [11].

The real-world workforce scheduling process starts from three entry points (see Figure 1). First, workload prediction (or demand forecasting) is the phase of determining the staffing levels - that is, how many employees are needed for each timeslot in the planning horizon. Shift generation is the phase of determining the shift structure, along with the activities to be carried out in particular shifts and the competences required for different shifts. Second, the HR master system provides necessary employee data, such as labor contract, work unit and working hours. Third, the competences and preferences of the employees are maintained by the workforce management system itself. These three entry points gather the required information for shifts and employees (see Figure 1).

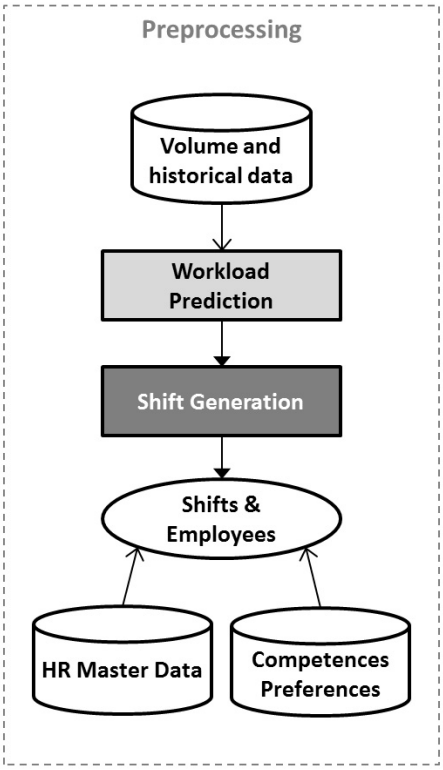

Fig. 1. The preprocessing phases of the real-world workforce scheduling process.

The nature of determining the amount and type of work to be done at any given time during the next planning horizon depends greatly on the nature of the job. Some form of workload prediction is called for if the workload is overly uncertain. Some examples of this are the calls incoming to a call center or the customer influx to a hospital. Modern workforce management systems (WFM) use simulation to calculate forecast values.

Shift generation transforms the determined workload into shifts. This includes deciding break times when applicable. Shift generation is essential especially in cases where the workload is not static. A basic shift generation problem

Manuscript received March 12, 2015.

Kimmo Nurmi and Jari Kyngäs are with the Satakunta University of Applied Sciences, Pori, Finland (phone: +358 44 710 3371, e-mail: cimmo.nurmi@samk.fi).

Nico Kyngäs is with the Numeron Ltd, Tampere, Finland (e-mail: nico.kyngas@numeron.com).

ISBN: 978-988-14047-0-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

includes a variable number of activities for each task in each timeslot. Activities may require competences. The most important optimization target is to match the shifts to the workload as accurately as possible. The shifts are generally created for each day separately, each shift corresponding to a single employee's competences and preferences.

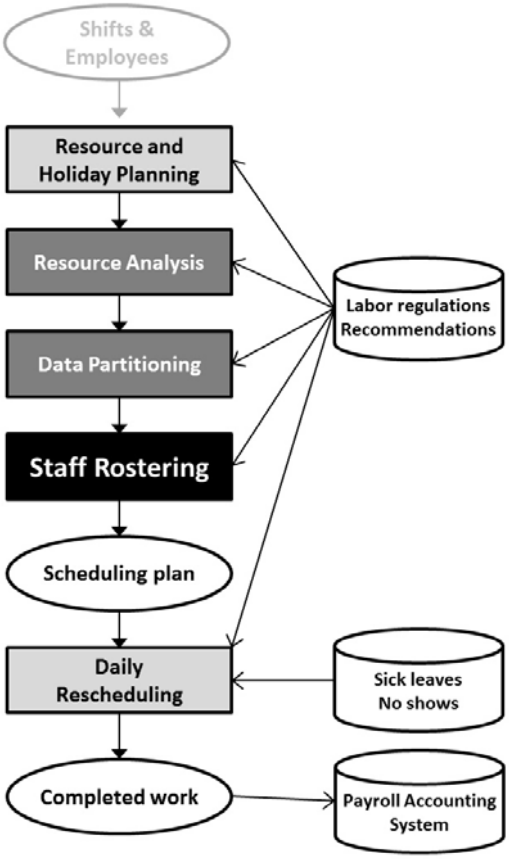

Fig. 2. The real-world workforce scheduling process.

Figure 2 shows the real-world workforce scheduling process. Future staffing requirements are carefully considered in resource and holiday planning. Holidays, training sessions and other short-term absences as well as long-term sick-leaves and forthcoming retirements have major impact to actual staff rostering. To see if there will be any chance of succeeding at matching the workforce with the shifts while adhering to the given constraints, a resource analysis should be run on the data. The analysis checks the balance between the shifts and the available employees.

The usefulness and utility of the optimized rosters depend more on the good-quality outcome of the preceding phases than the actual optimization result. However, when the input data for the staff rostering is valid and correct, it's possible to gain significant benefit in financial efficiency and employee satisfaction by using optimization.

Some real-world datasets are huge. They may consist of hundreds of employees with a corresponding number of jobs. In these cases it is probably computationally impossible to try to roster the whole set of employees at once. Therefore, we could partition the data into smaller units, roster them separately and assemble the units back together. Unfortunately, this is as difficult a problem as the staff rostering itself.

Staff rostering is the phase where the shifts are assigned to the employees. The length of the planning horizon is usually between two and six weeks. The most important constraints are employees' competences and preferences as well as the working and resting times, since these are laid down by the collective labor agreements and government regulations.

Unfortunately, the optimized staff rosters need to be changed. Daily rescheduling deals with ad hoc changes that are necessary due to sick leaves and other no-shows. The changes are usually carried out manually. Still, the system should suggest suitable substitutes considering the qualifications, employment contract, legal limitations and salaries. The goal is to find the most economical candidates.

Finally, the completed working times will be booked and made available for payroll accounting system. When necessary, the workload prediction or/and the shift generation phases may be restarted. In any case, the staff rostering will be restarted for the next planning horizon.

As stated earlier, most of the academic studies focus on staff rostering. Some academic researchers have even announced that their optimization methods are in commercial use [12-15]. Some real-world benchmark instances [16-19] exist to help researchers to test their optimization methods. To the best of our knowledge, no simple random test instances exist. The goal of this paper is to introduce a new set of benchmark instances for the core staff rostering problem. The test instances are simple enough to ensure that they will not remain unsolved by other researchers because of the complexity of the instance.

Section II describes the core staff rostering problem and introduces the necessary terminology. Section III introduces the benchmark instances. In Section IV we briefly describe our PEAST algorithm which is used to solve the benchmark instances. Section V sets out the computational results.

II. The Core Staff Rostering Problem

The staff rostering problem consists of assigning employees to shifts and tasks over a period of time. The *planning horizon* is the time interval over which the employees have to be scheduled. Each employee has a total working time that he/she has to work during the planning horizon. Furthermore, each employee has *competences* (qualifications and skills) that enable him/her to carry out certain tasks. Days are divided into working days (*days-on*) and rest days (*days-off*). Each day is divided into periods or timeslots. A *timeslot* is the smallest unit of time and the length of a timeslot determines the granularity of the schedule. A *shift* is a contiguous set of working hours and is defined by a day and a starting period on that day along with a shift length (the number of occupied timeslots) or shift time. Shifts are sometimes grouped into shift types, such as morning, day and night shifts. Each shift is composed of *tasks* and *breaks*. The sum of the length of a shift's tasks is called *working time*. A shift or a task may require the employee assigned to it to possess one or more competences. A sequence of working days with one shift each day is called a *work stretch*. A work schedule over the planning horizon for an employee is called a *roster*. A roster is a combination of shifts and days-off assignments that

ISBN: 978-988-14047-0-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

covers a fixed period of time. Figure 3 clarifies the terminology.

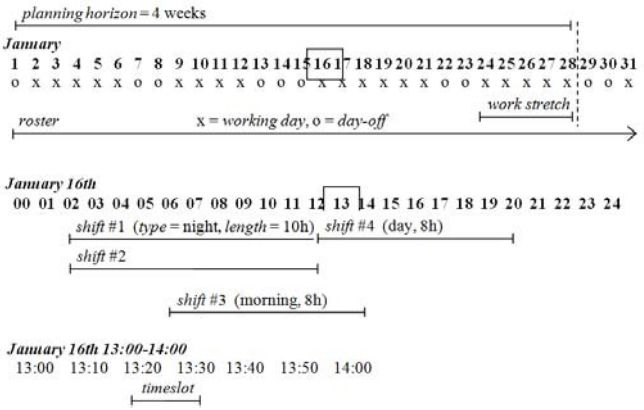

Fig. 3. Staff rostering terminology.

There are hundreds of staff rostering solutions commercially available and in widespread use. Only few of them include optimization. It is difficult to incorporate the experience and expertise of the personnel managers into a staff rostering system. Personnel managers have extremely valuable knowledge, experience and detailed understanding of their specific staffing problem, which will vary from company to company. To formalize this knowledge into constraints is not an easy task.

Dozens of optimization models for staff rostering have been introduced by academics (see [8-11]). These models present a wide variety of real-world constraints. The goal is to build up a solid foundation for staff rostering scenarios. The most important goal of the models is to minimize understaffing and overstaffing without violating work contracts and government regulations. Some of the models also pay attention to employee requests.

We model *the core staff rostering problem* as a constraint satisfaction problem. The model includes eight constraints that comprise the core of almost any staff rostering problem:

Hard constraints

- (H1) An employee cannot be assigned to more than one shift on a day and cannot be assigned to overlapping shifts
- (H2) Maximum of one employee can be assigned to a shift
- (H3) An employee must not work consecutively for more than the given number of days
- (H4) The given number of rest time between two shifts should be respected

Soft constraints

- (S1) Each employee's working time should be within the given limits (one violation for each starting 10 minutes below or above)
- (S2) Single days-off should be avoided (one violation for each such case)
- (S3) Single working days should be avoided (two violations for each such case)
- (S4) The given (minimum) number of free weekends, both Saturday and Sunday free, for each employee should be respected (one violation for each case below).

The constraints state that the task is to meet the daily staffing requirements without violating work contracts. The goal is to find a solution that has no hard constraint violations and that minimizes the sum of the soft constraint violations. Note, that single days-off and single working days at the start or at the end of the planning period are not calculated.

III. The Benchmark Instances

We believe that introducing simple benchmark instances of the core staff rostering problem will be very valuable to researchers. This enables a large number of researchers to find out how well their optimization methods work. The number of constraints in the current staff rostering benchmark instances is quite large and some of the models are also quite complicated.

Researchers quite often only solve one real-world case. The strength of random test instances is the ability to produce many problems with many different properties. Still, they should be sufficiently simple for each researcher to be able to use them in their test environment.

The No Free Lunch Theorem [20] states that all search algorithms that look for an optimum of some cost function perform exactly identically when averaged over all possible cost functions. As we know, this result implies three important conclusions:

1) If a search algorithm works well for a given problem type, it is not guaranteed to work as well for a different problem type
2) There exists no superior optimization method
3) If we are faced with a new problem area, we should first analyze the problem and then choose the most applicable optimization method, not vice versa.

However, there is still room for comparisons between different optimization methods, albeit not in the sense of finding the overall best one, but to find out on which problem areas a particular method is likely to be better than others. We invite the workforce scheduling community to try their methods on the core staff rostering problem.

Table I shows ten random test instances for the core staff rostering problem. The number of employees (E) varies between 50 and 100. The planning period (W) is 4, 6 or 8 weeks. The number of shifts (S) is 1000, 2000 or 2250. The length of a shift is a uniform random number on the interval given by L. The required working time of an employee is given by an interval, for example between 10000 and 11000 minutes. The intervals are randomly created based on the total number of working time required by the shifts. The average required working time of the employees is given by *T-avg*. The minimum and maximum intervals of the working times are given by *I-min* and *I-max*. The data for the test instances is available online [26].

Table II shows the parameters and weights of the constraints. The hard constraints *H1* and *H2* have no additional parameters. The maximum number of consecutive working days is given in *H3* and the minimum rest time in minutes between two shifts is given in *H4*. The

ISBN: 978-988-14047-0-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

weights of the soft constraints are given by *S1*, *S2*, *S3* and *S4*. The minimum number of free weekends for *S4* is given in parentheses.

TABLE I
BENCHMARK INSTANCES - DATA

ID	E	W	S	L	T-avg	I-min	I-max
01	100	4	2000	450 – 600	10499	25	514
02	50	8	2000	240 – 720	19211	307	1704
03	50	4	1000	460 – 500	9598	15	159
04	100	4	2000	440 – 520	9569	133	1840
05	100	4	2000	360 – 540	9093	25	340
06	50	8	2000	360 – 540	17806	110	1034
07	50	4	1000	450 – 550	9997	15	93
08	75	6	2250	100 – 700	11988	7	158
09	100	4	2000	100 – 700	8058	92	3107
10	75	6	2250	700 – 900	24556	23	2308

E = number of employees, *W* = number of weeks, *S* = number of shifts, *L* = shift length interval, *T-avg* = average required working time, *I-min* and *I-max* = minimum and maximum interval of the working time

TABLE II
BENCHMARK INSTANCES - CONSTRAINTS

ID	H1	H2	H3	H4	S1	S2	S3	S4
01	on	on	6	540	2	2	1	3 (1 week)
02	on	on	8	720	2	2	1	3 (2 weeks)
03	on	on	6	720	2	2	1	3 (2)
04	on	on	6	540	2	2	1	3 (1)
05	on	on	5	510	2	2	1	3 (2)
06	on	on	8	540	2	2	1	2 (3)
07	on	on	5	720	2	1	1	1 (1)
08	on	on	5	700	2	3	1	3 (3)
09	on	on	5	1000	3	1	2	1 (3)
10	on	on	6	360	3	1	3	3 (4)

H3 = maximum number of consecutive working days, *H4* = minimum rest time in minutes between two shifts, *SX* = weight of the soft constraint *X*

IV. THE PEAST ALGORITHM

This section briefly describes the PEAST algorithm which is used to solve the benchmark instances presented in Section III. The PEAST algorithm has been used to solve several real-world scheduling problems (see eg. [20-24]) and it is in industrial use.

The pseudo-code of the algorithm is given in Figure 4. The algorithm is a population-based local search method. In every c^{th} iteration, the least fit schedule is replaced with a clone of the fittest individual. This operation is completely irrespective of the globally fittest schedule (*best_S*) found by that time in the search process.

The heart of the algorithm is the local search operator called GHCM (greedy hill-climbing mutation). It extends the basic hill-climbing step to generate a sequence of moves in one step, leading from one solution to another.

The PEAST algorithm applies a number of shuffling operators to perturb a solution into a potentially worse solution in order to escape from local optima. The operators are called after a given number of iterations have passed. For example, in staff rostering two random employees are selected and all their jobs are swapped.

The algorithm avoids staying stuck in the same areas of the search space using tabu search and the refined simulated annealing method. A tabu list is used to prevent reverse order moves in a single application of the GHCM operator. The simulated annealing refinement is used to decide whether or not to commit to a sequence of moves in the GHCM operator.

```
Input the population size n, the iteration limit t,
the cloning interval c, the shuffling interval s and
the ADAGEN update interval a
Generate a random population of schedules S_i for i = 1, ..., n
Set best_S = null and iteration = 1
WHILE iteration ≤ t
  k = 1
  WHILE k ≤ n
    Apply GHCM to schedule S_k to get a new schedule
    IF Cost(S_k) < Cost(best_S) THEN Set best_S = S_k
    k = k + 1
  END REPEAT
  Update the simulated annealing framework
  IF iteration ≡ 0 (mod c) THEN
    Replace the worst schedule with the best one
  IF iteration ≡ 0 (mod s) THEN
    Apply shuffling operators
  IF iteration ≡ 0 (mod a) THEN
    Update the ADAGEN framework
  iteration = iteration + 1
END WHILE
Output best_S
```

Fig. 4. The pseudo-code of the PEAST algorithm.

The PEAST algorithm uses the adaptive genetic penalty method (ADAGEN) to solve multi-objective problems such as the constraint satisfaction problem. The method assigns dynamic weights to the hard constraints based on the search trajectory and the constant weights assigned to the soft constraints. The soft constraints are assigned fixed weights according to their significance. The significance is given by the problem owner (end-user).

The acronym PEAST stems from the methods used: Population, Ejection, Annealing, Shuffling and Tabu. To the best of our knowledge, the heart of the algorithm, the GHCM operator, is one of a kind. The same applies to our implementation of the shuffling operators, the simulated annealing refinement and the penalty method. For the detailed discussion of the PEAST algorithm and its components we refer to [25].

V. COMPUTATIONAL RESULTS

This section summarizes the best solutions we have found for the benchmark instances. We solved the instances using the PEAST algorithm described in Section IV. We used exactly the same version and parameters of the algorithm that are in use in the real-world workforce management system.

We found a feasible solution to all instances. This was actually not difficult. Table III shows our best solutions from ten runs. The instances *05* and *10* were the most difficult to solve. The most difficult task was to minimize the number of single days-off. We believe that the number of violations for the soft constraint S1 for the instances *05*, *06* and *08* is optimal. We are quite certain that better overall solutions for all the test instances exist.

ISBN: 978-988-14047-0-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE III
BENCHMARK INSTANCES – BEST SOLUTIONS

ID	H1	H2	H3	H4	S1	S2	S3	S4	Best
01	0	0	0	0	0	90	0	18	234
02	0	0	0	0	0	31	0	13	101
03	0	0	0	0	0	36	0	63	261
04	0	0	0	0	0	56	0	0	112
05	0	0	0	0	24	264	13	106	907
06	0	0	0	0	683	37	1	0	1441
07	0	0	0	0	0	93	0	0	93
08	0	0	0	0	2543	167	0	134	5989
09	0	0	0	0	0	161	0	221	382
10	0	0	0	0	0	63	5	0	78

The cells give the number of violations for each constraint

VI. CONCLUSIONS

We defined the core staff rostering problem as a constraint satisfaction problem that includes eight constraints that comprise the core of almost any staff rostering problem. We introduced ten random test instances for the problem.

We solved the instances using the PEAST algorithm. The workforce scheduling community is invited to challenge our solutions. We hope that the instances will help researchers to test the value of their optimization methods. The instances are available online.

REFERENCES

[1] M.R. Garey and D.S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[2] J. Tien and A. Kamiyama, "On Manpower Scheduling Algorithms", *SIAM Rev.* 24 (3), 1982, pp. 275–287.

[3] H.C. Lau: "On the Complexity of Manpower Shift Scheduling", *Computers and Operations Research* 23(1), 1996, pp. 93-102.

[4] D. Marx: "Graph coloring problems and their applications in scheduling", *Periodica Polytechnica* Ser. El. Eng. 48, 2004, pp. 5–10.

[5] L. Di Gaspero, J. Gärtner, G. Kortsarz, N. Musliu, A. Schaerf and W. Slany: "The minimum shift design problem", *Annals of Operations Research,* 155(1), 2007, pp. 79-105.

[6] M. J. Brusco and T. R. Johns: "A sequential integer programming method for discontinuous labor tour scheduling", *European Journal of Operational Research* 95, 1996, pp.537-548.

[7] G.B. Dantzig: "A comment on Edie's traffic delays at toll booths", *Operations Research* 2, 1954, pp. 339–341.

[8] H.K. Alfares: "Survey, categorization and comparison of recent tour scheduling literature", *Annals of Operations Research* 127, 2004, pp. 145-175.

[9] A.T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier: "Staff scheduling and rostering: A review of applications, methods and models", *European Journal of Operational Research* 153 (1), 2004, pp. 3-27.

[10] A. Meisels and A. Schaerf: "Modelling and solving employee timetabling problems", *Annals of Mathematics and Artificial Intelligence* 39, 2003, pp. 41-59.

[11] P. De Causmaecker and G. Vanden Berghe: "A categorisation of nurse rostering problems", *Journal of Scheduling* 14(1), 2011, pp. 3-16.

[12] E. Burke, P. De Causmaecker, S. Petrovic and G. Vanden Berghe: "Metaheuristics for Handling Time Interval Coverage Constraints in Nurse Scheduling", *Applied Artificial Intelligence* 20, 2006, pp. 743-766.

[13] B. Bilgin, P. De Causmaecker, B. Rossie and G. Vanden Berghe: "Local Search Neighbourhoods to Deal with a Novel Nurse Rostering Model", In *Proc. of the 7th Int. Conf. on the Practice and Theory of Automated Timetabling*, Montréal, Canada, 2008.

[14] G.R. Beddoe, S. Petrovic and J. Li: "A Hybrid Metaheuristic Case-based Reasoning System for Nurse Rostering". *Journal of Scheduling 12*, 2009, pp. 99-119.

[15] N. Kyngäs, K. Nurmi and J. Kyngäs, "Workforce Scheduling Using the PEAST algorithm", in Ao, Sio-Iong (ed.): *IAENG Transactions on Engineering Technologies, Lecture Notes in Electrical Engineering* 275, Springer, USA, 2014, pp 359-372.

[16] T. Curtois: "Staff Rostering Benchmark Data Sets" [Online]. Available: http://www.cs.nott.ac.uk/~tec/NRP/, (Last access January 2015).

[17] B. Bilgin: "Nurse rostering benchmark datasets" [Online], Available: http://allserv.kahosl.be/~pieter/nurserostering.html, (Last access January 2015).

[18] P. De Causmaecker: "The First International Nurse Rostering Competition" [Online], Available: https://www.kuleuven-kulak.be/nrpcompetition, (Last access January 2015).

[19] N.T.T. Dang and P. De Causmaecker: "The Second International Nurse Rostering Competition" [Online], Available: http://mobiz.vives.be/inrc2/, (Last access January 2015).

[20] N. Kyngäs, K. Nurmi and J. Kyngäs: "Solving the person-based multitask shift generation problem with breaks", In *Proc. of the 5th International Conference On Modeling, Simulation And Applied Optimization*, Hammamet, Tunis, 2013, pp. 1-8.

[21] N. Kyngäs, K. Nurmi and J. Kyngäs: "Optimizing Large-Scale Staff Rostering Instances", *Lecture Notes in Engineering and Computer Science: Proceedings of The International MultiConference of Engineers and Computer Scientists*, Hong Kong, 2012, pp. 1524-1531.

[22] K. Nurmi, J. Kyngäs, Zheng-Yun Zhuang and N. Kyngäs: "Optimized Workforce Scheduling in Bus Transit Companies", in *Proc of the 4th Global Congress on Intelligent Systems*, Hong Kong, 2013, pp. 301-305.

[23] K. Nurmi and J. Kyngäs: "A Conversion Scheme for Turning a Curriculum-based Timetabling Problem into a School Timetabling Problem", in *Proc of the 7th Conference on the Practice and Theory of Automated Timetabling (PATAT)*, Montreal, Canada, 2008.

[24] K. Nurmi, J. Kyngäs, D. Goossens and N. Kyngäs: "Scheduling a Professional Sports League using the PEAST Algorithm", *Lecture Notes in Engineering and Computer Science: Proceedings of The International MultiConference of Engineers and Computer Scientists*, Hong Kong, 2014, pp. 1176-1182.

[25] N. Kyngäs, K. Nurmi and J. Kyngäs: "Crucial Components of the PEAST Algorithm in Solving Real-World Scheduling Problems", *Journal of Lecture Notes on Software Engineering* 1(3), 2013, pp. 230-236.

[26] K. Nurmi: "The Core Staff Rostering Problem – Benchmark Instances" [Online]. Available: http://web.samk.fi/public/tkiy/CSP/, (Last access March 2015).

ISBN: 978-988-14047-0-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)