Domain Transformation using Greedy Algorithm in Nurse Scheduling

Geetha Baskaran[1], Andrzej Bargiela[2], and Rong Qu[3]

***Abstract*—**The nurse scheduling problem (NSP) is a complex optimisation problem of allocating nurses to duty rosters in hospitals. The objective is usually to ensure that there are always sufficient nurses on duty, while taking into account individual preferences with respect to work patterns, requests for leave and financial restrictions, in such a way that all employees are treated considerately and fairly. In this paper, we extend our domain transformation-based approach to nurse scheduling. Domain transformation is an approach to solving complex problems that involves a judicious simplification of the original problem, a solution to the simplified problem and finally a transformation of this solution to compensate for the introduced simplifications. The approach we use involves information granulation of shift types that transforms the problem into a smaller solution domain. Next, schedules derived from smaller problem domains are then converted into the original problem domain. This includes a consideration of the constraints that were not represented in the smaller domain. The problem is solved using a greedy algorithm. The approach is general and can be applied within a wide range of benchmark instances. These benchmark instances have been derived from published datasets as well as from our industry collaborators. The greedy algorithm outperforms in most cases when tested with the different demands and number of nurses. The results facilitated the development of a cost–benefit analysis across different levels of staffing.

Keywords*—domain transformation, nurse scheduling, greedy hill climbing.*

I. Introduction

THE nurse scheduling problem (NSP) has been a widely considered topic for many years due to ongoing nurse shortages and the associated sophisticated and challenging real-world issues in nurse management systems. Part of the staff scheduling problem is to determine the times at which shifts are allocated to each member of the nursing staff [21]. Nurse scheduling is defined as assigning specific nurses to work at particular shifts [30]. Automatic approaches have significant benefits in saving administrative staff time and also generally improving the quality of the schedules produced. However, until recently, scheduling has often been done manually, which is usually a very time-consuming task. In addition, it seldom gives the best quality results [17]. As nurses are needed in hospital wards for 24 hours a day, their work schedules have to be arranged effectively. Nurse scheduling aims to create a more systematic approach to the assignment of nurses' shifts, and to scheduling within fixed time periods. Nurse scheduling also aims to ensure that high-quality services are consistently provided [7; 5; 8].

These issues have led to a number of alternative attempts to solve real-world NSPs. Since the 1980s, artificial intelligence methods for nurse rostering, such as constraint programming [20], expert systems [25] and knowledge-based systems [16] have been investigated with some success. Since the 1990s, many researchers have tackled the problem with meta-heuristic methods, which include simulated annealing [28], variable neighbourhood searching [8], tabu searching [27] and evolutionary methods [9; 19]. In more recent years, there has been increasing interest in the study of mathematical programming-based heuristics [22; 23; 24] and the use of hyper-heuristics [11; 31] to address the problem [11; 13].

In the NSP, there are both hard and soft constraints. Hard constraints must be considered at all times and may render the schedule unworkable. Soft constraints operate to determine the quality of the solution. In this sense, soft constraints are not necessary, but should be addressed as often as possible. Nevertheless, to achieve a schedule that addresses all of the hard constraints, breaking some of the soft rules is necessary. A weight is allocated for each soft constraint that reflects its worth. The objective of nurse scheduling is to find a schedule that satisfies all of the hard constraints and minimises the degree to which the soft constraints are violated.

In this study, we present an alternative method for tackling a large, real-world NSP using a greedy algorithm across a large collection of diverse scheduling benchmark instances. In this approach, the hospital is supplied with detailed information about the schedule that they can use to make an objective selection. We used the domain transformation method to approach the problem of cost effectiveness in scheduling nurses. [This approach was introduced practically by [15] to demonstrate the use of information granulation methodology [1; 2] for the generation of multiple feasible low-cost rosters.] We then evaluated the resulting rosters.

II. Nurse Scheduling Problem using Domain Transformation

The NSP is defined as a problem of assigning each nurse a specific shift within a pre-defined scheduling horizon. It is subject to hard constraints that originate from contractual

Geetha Baskaran[1] is with the School of Computer Science, University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih, Malaysia (e-mail: Geetha.baskaran@nottingham.edu.my).

Andrzej Bargiela[2], is with the School of Computer Science, University of Nottingham, Jubilee campus & Krakow Technical University, Krakow, Poland (e-meail: abb@cs.nott.ac.uk).

Rong Qu[3] is with the School of Computer Science, University of Nottingham, Jubilee campus, Nottingham, NG8 1BB, UK (e-mail: rxq@cs.nott.ac.uk).

agreements, legal requirements and local good practice. Any schedule that satisfies these constraints is referred to as a feasible schedule. Satisfying hard constraints is only a starting point for the construction of a good-quality schedule. The degree of satisfaction of additional constraints reflecting staff preferences for allocation of specific shifts provides a measure of the quality of the schedule.

We produce feasible scheduling by generating pattern sequences [14]. The scheduling problem is now changed to a problem of scheduling patterns. The number of patterns that needs to be considered in domain transformation determines the computational gain.

Our *domain transformation* approach can be summarised as a three-stage process:

I. conversion of the problem from the original edlNR domain into a problem in the ***D***NR domain
II. solution of the problem in the ***D***NR domain
III. conversion of the ***D***NR solution into a solution in the original edlNR domain.

We achieve gains in performance in the patterns of the N-shifts and reduced number of nurses that needs to be allocated into D-shifts.

The approach relies on well-justified simplification of the original problem. The problem is divided into smaller sub-problems in a systematic way that can be reproduced. This approach avoids a random search. Other methods failed to reproduce results, have inconsistent performance and work only on selected datasets while failing on others. The previous state-of-the-art approach did not use information granulation (domain transformation approach (DTA)), and was forced to deal with data checking and cross-referencing issues. We have investigated the optimum balance between the staffing levels of the ward and the ability to achieve good-quality schedules.

A. Benchmark Instances

To validate our algorithms and encourage more competition and collaboration between researchers addressing scheduling, we have built a collection of diverse and challenging benchmark instances. The collection has grown over several years and been sourced from 13 different countries, and the majority of the datasets are based on real-world scheduling scenarios. Table 1 lists these instances. They vary in the length of the planning horizon, the number of employees, the number of shift types and the number of skills required. Each instance also varies in the number, priority and type of constraints as well as the objectives present. The objectives were set by the organisation that provided the data. For example, some prefer to minimise overstaffing whereas others prefer to maximise staff satisfaction by setting a higher importance weighting for those objectives instead.

TABLE 1
BENCHMARK INSTANCES

Instance	Staff	Shift types	Length (days)	Skill types	Best known	Ref
Musa	11	1	14	3	175	[4]
GPost	8	2	28	1	5	
GPost-B	8	2	28	1	3	
Ozkarahan	14	2	7	2	0	[25]
Millar-2Shift-Data1	8	2	14	1	0	[3]
Millar-2Shift-Data1.1	8	2	14	1	0	[3]
Azaiez	13	2	28	2	0	[29]
WHPP	30	3	14	1	5	[18]
Valouxis-1	16	3	28	1	20	[6]
Ikegami-2Shift-Data1	28	2	30	9	0	[3]
Ikegami-3Shift-Data1	25	3	30	8	2	[3]
Ikegami-3Shift-Data1.1	25	3	30	8	3	[3]
Ikegami-3Shift-Data1.2	25	3	30	8	3	[3]
ORTEC01	16	4	31	1	270	[12]
ORTEC02	16	4	31	1	270	[12]
QMC-1	19	8	28	1	13	
QMC-2	19	3	28	3	29	
SINTEF	24	5	21	1	0	

The instances are available for download from http://www.cs.nott.ac.uk/~tec/NRP/.

III. SOLUTION APPROACH WITH GREEDY ALGORITHM

Domain transformation is proposed to solve the problem. The initial solution is computed by means of a greedy algorithm. A complete solution for this problem is defined for each day of the month and for each nurse on the corresponding shift.

A. Get the First Week Cost Groups

Once the patterns are generated [14], we use them to schedule the one week in the ***D***NR domain. We group the patterns in three categories: cost 0, cost 5 and cost 10 according to the full-time and part-time nurses' patterns. We list all the night patterns (which may or may not include day patterns) and *only* day patterns that contain the day-only patterns. We also list the schedule set that contains the one-week schedule and nurse information. Schedule set is tied to nurse, so the number of elements in a schedule set is the same as the number of nurses. Also, the schedule set contains information related to nurses; such as costs, constraint violations and hours.

B. User Interface

The 'Create patterns button' runs the generating pattern code for creating patterns. The 'Create schedule button' reads input.txt for configuration and reads a common patterns document and stores that information in a database. Then it creates schedules. When the button is pressed, a layout such as in Figure 1 is created.

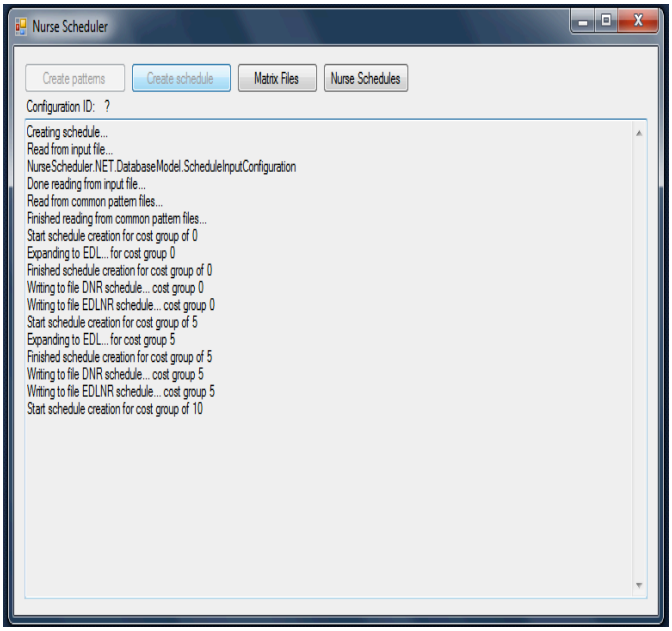

Fig. 1 Layout of create schedule

The 'Matrix files' history shown in Figure 2, illustrates the schedules generated by weeks stored in a MatrixFiles table created with the previous button. All patterns used in this process are saved there for all weeks and all costs used. It does not use matrices or integer programming.

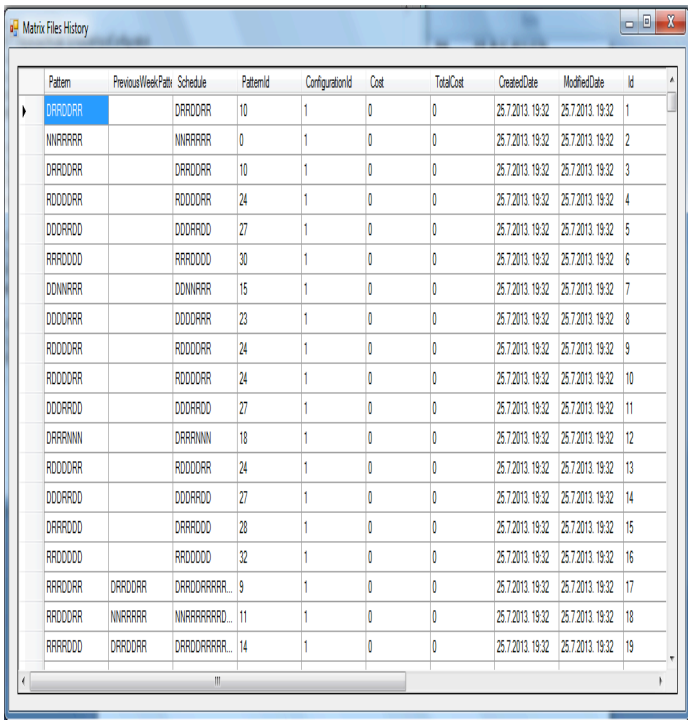

Fig. 2 Layout of schedule generated by weeks

The 'Nurse schedules' button creates the final nurse schedules made with ***D***NR and edlNR for all three cost groups as shown in Figure 3.

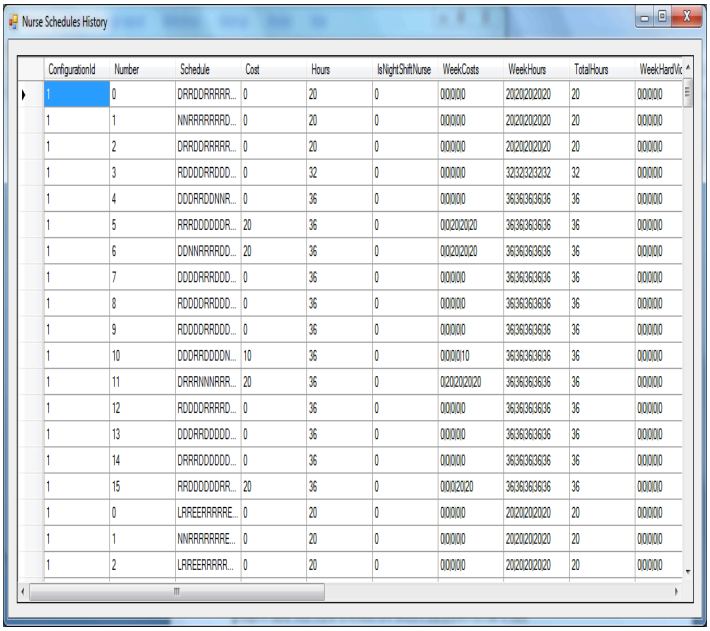

Fig. 3 Layout of schedule generated in both domains

A new configuration ID is assigned each time 'create schedule' is started. This ID can be used as a reference number in tables with nurse schedules, as shown in Figure 4.

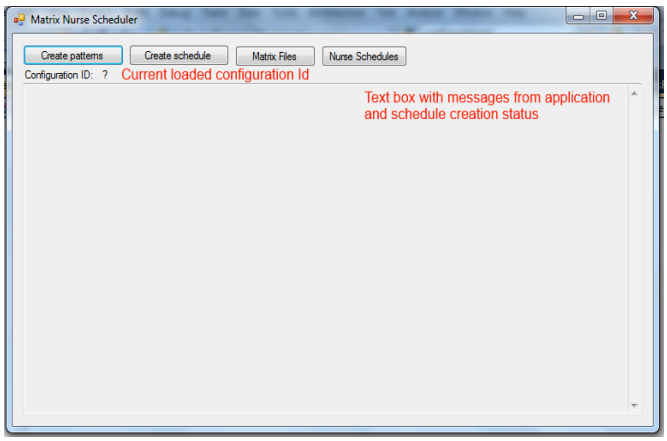

Fig. 4 Layout of interface parts and the placement of configuration ID

C. Workflow for Generating first Week Schedule

Associated patterns based on zero-cost patterns with nurses are based on full-time or part-time schedules. This is called the schedule set, which is stored in a vector object (array).

As it is difficult to satisfy the night shifts, we place the night patterns in an array. Indirectly, the night shift patterns also satisfy a few other constraints. A greedy algorithm examines all the days trying to guarantee the requested coverage for each shift. This is done by selecting, for each given shift, the best nurse to be assigned to that shift. So, day-only patterns are placed in the array. Here, demand is calculated to check the remaining number of day and night shifts to be filled per day.

The remaining demand is calculated by looping all nurses for each day, counting the number of shifts and subtracting them from the total allowed demand for each type of shift. We use two nested loops (nurses and days) for counting shifts

used in total for a day. Then, when calculating the difference, *if*s (for shift types) and *for*s (for days) are used, separating shift types when calculating the difference, per day. For example, if for some days D is allowed (***D***NR mode), demand is 9, four nurses only have D shift and remaining demand is 5. A similar approach is used for other shift types. Nurses are looped to assign a pattern to the nurse and meaning to the schedule set. For the first week, nurses are considered to be carriers of the schedule set.

1.0 The nurse assignment method is invoked where:
- 1.1 we check if night shift is allowed
- 1.2 all day-only patterns are looped (*for* loop)
  - 1.2.1 patterns are checked and validated regarding demand (length of pattern is checked depending on number of hours the nurse works, which cannot be more than six days in one week)
  - 1.2.2 pattern value is calculated (number of working shifts in pattern)
- 1.3 go back to 1.0 until the end of the number of day-only patterns
- 1.4 all night shift patterns are looped, if nurse allows night shifts (*for* loop)
  - 1.4.1 patterns are checked and validated regarding demand (length of pattern is checked depending on number of hours the nurse works, which cannot be more than six days in one week)
  - 1.4.2 pattern value is calculated (number of working shifts in pattern)
- 1.5 it is checked which pattern is the best fit
  - 1.5.1 if night shift is not allowed, the best day-only-valued pattern that fits the demand is assigned to the nurse
  - 1.5.2 if night shift is allowed, the best night shift-valued pattern is assigned if existing or not used, otherwise the day pattern is assigned
  - 1.5.3 a pattern is assigned, if possible night pattern, otherwise, day pattern
- 1.6 if too many night shifts are present in the schedule and exceed the demand, excessive night shifts are replaced with R (free days).
- 1.7 end of nurse assignment method

Later, a check is made of whether the demand is met. If not, the nurse repair function is called to repair the missing demand by replacing patterns with other day or night patterns. A check is also made of whether the patterns are correct. The function is run repeatedly. Later, the number of demand-remaining patterns is calculated.

2.0 For each day in a week:
- 2.1 if more than the needed day shifts are assigned, fix pattern method is called
  - 2.1.1 the fix pattern replaces someday shifts with a free day
  - 2.1.2 pattern validity is checked
- 2.2 If more than the needed night shifts are assigned fix pattern method is called
  - 2.2.1 the fix pattern replaces some night shifts with a free day
  - 2.2.2 pattern validity is checked

3.0 number of remaining demand patterns is calculated

4.0 for each day in a week

4.1 if less than the needed day shifts are assigned, fix pattern method is called
- 4.1.1 the fix pattern replaces someday shifts with a free day
- 4.1.2 pattern validity is checked

4.2 if less than the needed night shifts are assigned, fix pattern method is called
- 4.2.1 fix pattern replaces some night shifts with a free day
- 4.2.2 pattern validity is checked.

In simple terms, first we use a greedy algorithm to create a schedule. Then we try to fit improve patterns in the schedule using the nurse repair function in a recursive fashion. The function is limited by time, so it will not try to find a better long-term pattern than what is found as optimal number of seconds while checking the results with experiment. After that, the fix pattern function is used to sort out demand, removing shifts from the schedule when the demand is overbooked. For the first week we use zero-cost patterns. Figure 5 provides a graphical illustration of the generation of the week one schedule.

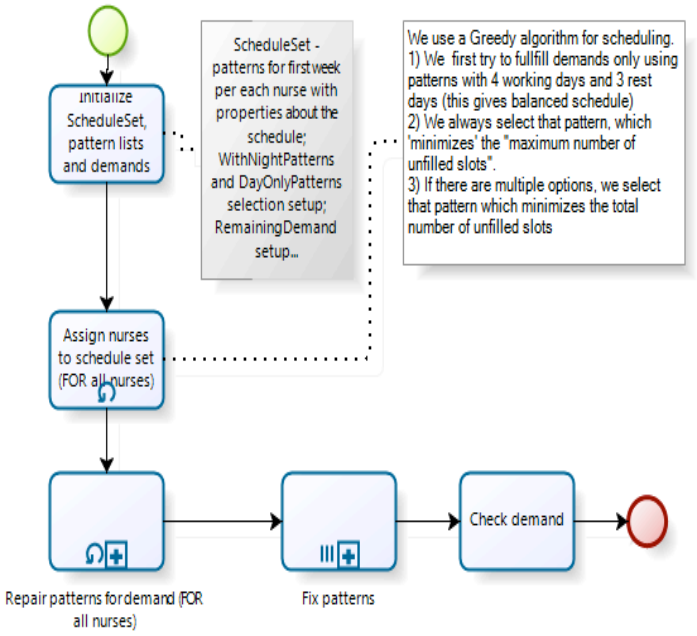

Fig. 5 The process of generating a Week one schedule

D. Workflow for Generating the N Week Schedule

To expand the schedule to N weeks, matrix base subsets are used to set combinations of nurses and patterns that can or cannot be used. First, the subset is based on pattern validity inside the first *for* loop. In the following *for* loop, it is set which nurses will and will not use night patterns. This function uses similar methods as for the first week, but with changes implemented to add checks of the current week's schedule with previous weeks. Further, it calculates the costs and violations. Last, it corrects the schedule in relation to the demand. The changes incorporated in week one are at 1.2.1

and 1.4.1. The patterns are checked and validated regarding cost and violations, where zero cost and zero violations are allowed. There is also some loop added in 1.5, where now in week N we name 5.5.

5.5 Which pattern is the best fit is checked
5.5.1 if no pattern is the best fit, we take a similar approach as for 1.2 but with increased cost
5.5.1.1 all day-only patterns are looped, and patterns are checked and validated regarding cost and violations, allowing up to 20 cost and zero violations. The same is done with night patterns. Pattern value is retrieved.
5.5.1.2 if the best fit is not found after this, we exit the assign nurse function.
5.5.2 if night shift is not allowed, the best day-only-valued pattern is assigned to nurses that fit the demand. We also check the cost of the pattern inside the schedule and violations, where zero cost is allowed.
5.5.3 if night shift is allowed, the best night shift-valued pattern is assigned if existing or not used; otherwise the day pattern is assigned. We also check the cost of the pattern inside the schedule, and violations, where zero cost is allowed.
5.5.4 a pattern is assigned: if possible night pattern, otherwise, day pattern

Again, a check is made of whether the demand is met, and of all the other steps as in week one generation. The difference here is the check is made for the cost, where zero cost and zero violation is allowed. Once more, the number of -remaining demand patterns is calculated as shown in 2.0, onwards to 4.2.2. Here at each *if* selection before the pattern validity is checked, costs and violations are checked. Between 0 and 20 cost patterns are allowed. This is because patterns formed together can incur costs. Thus, the pattern is checked to see whether it forms a cost within the schedule and that it has zero hard violations. First it is checked whether the schedule can be made with zero cost, and if not, then cost up to 20 is allowed per nurse.

E. Workflow Convert **D***NR Domain to edlNR Domain*

At this stage, we convert the ***D***NR domain to the original edlNR domain. Here the EDL patterns are involved. We look for the next series with a day pattern and check its position of next day pattern, night pattern and series length (this attempt is based on solving series). First, we check if the pattern ends with a night series. If yes, we place 2–3 Ls or Es before N shift series, if this is possible with demand. We keep the Ds where it is not possible to replace with L or E, depending on demand. Besides, if the series is first in the schedule, we allow one day length of miniseries (of Ls or Es); or else if the pattern does not end with a night series,we place Es at the beginning of short series with length 2–3, depending on demand. In addition, we place Ls at the end of the series with length of 2–3 and keep Ds where it is not possible to replace with L or E, depending on demand. We then look for the next series. At this juncture we always check the demand. Once that is done, we backtrack all the nurses. We loop days and check missing demand for the day and for E or L. Then we place E or L for found D shift. Again, we check the demand and EDL costs and violations.

After this procedure, it is checked if the schedule is required for incomplete weeks. So if the schedule begins later than Monday and ends earlier than Sunday, the extra shifts are removed. Cost is recalculated with the extra days removed from the schedule, so it is possible that the schedule cost is lower than for full weeks. This whole process is best illustrated in Figure 6.

IV. Computational Results

For our approach, we did the testing using the Intel Pentium Dual Core T4500 2.3 GHz x64 PC with 4 GB RAM under Windows 8 Pro x64 was used. We have also used the database engine which is SQL Server Compact 3.5. The results obtained by solving the Greedy are presented in the table 2.

The Greedy method was able to solve most of the instances to optimality, but the computation time varied from less than 0.1 second to 1.39 minutes in the case of the hardest instance. In comparison with the best known result, we have achieved a new result for GPost-B with cost, two in 15 seconds and WHPP zero-cost in 17 seconds. One result for large instances outperformed the other examples, ORTEC, which appears significantly better than the best results achieved in the existing literature. Our approach achieved 130 costs within 85 seconds for ORTEC01 and 99 seconds for ORTEC02. Overall, our results are equal to the best known cost. However, 50 per cent of the result has a slower computation score when compared to Burke et al.'s results (2014).

V. Conclusion

We have presented a novel, information granulation-based formulation of the NSP and have solved it using a greedy algorithm. The results show that the greedy method can solve some instances very effectively. In other instances, the time and resource requirements may be restrictive. However, through the development of new ideas it may be possible to further improve the performance. The domain transformation approach uses a number of novel ideas that we believe are general enough to be adapted to other problem domains. All examples tested were modelled using a generic model. Taken as a whole, the proposed approach has a number of distinct advantages. The greedy algorithm is easy to implement, which ensures a good solution is obtained in real time.

References

[1] A. Bargiela and W. Pedrycz, "Granular Computing –An Introduction", Kluwer Academic Publishers. 2002. http://dx.doi.org/10.1007/978-1-4615-1033-8.

[2] A. Bargiela, and W. Pedrycz, "Toward a theory of Granular Computing for human-centred information processing", IEEE Trans. On Fuzzy Systems. 16(2): 320-330. http: //dx.doi.org/10.1109/TFUZZ.2007.905 912. 2008.

[3] A. Ikegami and A. Niwa, A Subproblem-centric Model and Approach to the Nurse Scheduling Problem, Mathematical Programming 97(3) (2003) 517-541.

[4] A. Musa and U. Saxena, Scheduling nurses using goal-programming techniques, IIE transactions 16 (1984) 216-221.

[5] B., H. Cheang, A. Li, B. Lim, Rodrigues, "Nurse rostering problems – a bibliographic survey" European Journal of Operational Research 151 447–460, 2003.

[6] C. Valouxis and E. Housos, Hybrid optimization techniques for the workshift and rest assignment of nursing personnel, Artificial Intelligence in Medicine 20(2) (2000) 155-175.
[7] D. Sitompul and S. Randhawa," Nurse Rostering Models: A State-of-the-Art Review". *Journal of the Society of Health Systems*. 2: 62-72. 1990
[8] E.K. Burke, P. De Causmaecker, G. Vanden Berghe, H. Van Landeghem., "The state of the art of nurse rostering" Journal of Scheduling 7 441–499, 2004
[9] E.K. Burke, P. Cowling, P. De Causmaecker, G. Vanden Berghe. "A memetic approach to the nurse rostering problem" Applied Intelligence 15 199–214, 2001.
[10] E.K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe. "Variable neighbourhood search for nurse rostering problems" M.G.C. Resende, J.P. De Sousa, eds. Metaheuristics: Computer Decision-Making (Combinatorial Optimization Book Series). Kluwer, Chapter 7. 153–172, 2004.
[11] E.K. Burke, G. Kendall, J. Newall, E Hart, P. Ross, S. Schulenburg. "Hyper-heuristics: an emerging direction in modern search technology" F. Glover, G. Kochenberger, eds. Handbook of Meta-Heuristics. Kluwer, Chapter 16. 451–470, 2003.
[12] E.K. Burke, T. Curtois, G. Post, R. Qu, and B. Veltman, A Hybrid Heuristic Ordering and Variable Neighbourhood Search for the Nurse Rostering Problem, European Journal of Operational Research 188(2) (2008) 330-341.
[13] E. Özcan, "Memetic algorithms for nurse rostering", Proceedings of the 20th International Symposium on Computer and Information Sciences. Springer Lecture Notes in Computer Science Volume 3733 482-492, 2005.
[14] G. Baskaran, A. Bargiela, R. Qu,. "Hierarchical method for nurse rostering based on granular pre-processing of constraints", *Proc. 23rd European Conference on Modelling and Simulation, ECMS*, Madrid, Spain, pp.855-861, June 2009, (doi: 10.7148/2009-0855-0861).
[15] G. Baskaran, , A. Bargiela, R. Qu, "A Study of Cost Effective Scheduling of Nurses Based on the Domain Transformation Method", Proc. 27th European Conference on Modelling and Simulation, ECMS, Alesund, Norway, pp.309-314, May 2013, (doi: 10.7148/2013-0309).
[16] G. Beddoe, S. Petrovic, "Selecting and weighting features using a genetic algorithm in a case-based reasoning approach to personnel rostering" (to appear) European Journal of Operational Research. 2006.
[17] H. Bouarab, S. Champalle, M. Dagenais, N. Lahrichi, A. Legrain, and M. Taobane,"Nurse Sceduling: From Theoretical Modeling to Practical Resolution", (2010). (Resource Optimization in Healthcare)
[18] H. Li, A. Lim, and B. Rodrigues. A Hybrid AI Approach for Nurse Rostering Problem. in Proceedings of the 2003 ACM symposium on Applied computing. 2003. pp. 730-735.
[19] H. Kawanaka, K. Yamamoto, T. Yoshikawa, T. Shinigi, S. Tsuruoka. "Genetic algorithm with the constraints for nurse scheduling problem" Proceedings of Congress on Evolutionary Computation. 1123-1130, 2001.
[20] H. Meyer auf'm Hofe, "Solving rostering tasks as constraint optimization" E.K. Burke, W. Erben, eds. Practice and Theory of Automated Timetabling, 3rd International Conference. Springer Lecture Notes in Computer Science Volume 2079 191–212, 2001.
[21] H. Rowland, & B. Rowland, "*Nursing administration handbook*", (1997). 4th ed. An Aspen Publication. USA
[22] J. Bard, H.W. Purnomo. "Preference scheduling for nurses using column generation" European Journal of Operational Research 164 510–534, 2006.
[23] J. Bard, H.W. Purnomo, "A cyclic preference scheduling of nurses using a Lagrangian based heuristic" Journal of Scheduling 10 5–23, 2007.
[24] J. Beliën, E.L. Demeulemeester. "Building cyclic master surgery schedules with leveled resulting bed occupancy" European Journal of Operational Research 176 1185-1204, 2006.
[25] Z.G. Chen, T. Yeung. "Hybrid expert system approach to nurse scheduling. Computers in Nursing" 183–192, 1993.
[26] J.L. Gino, A. Mobasher, J.C. Murray, 2012. "Multi-objective Nurse Scheduling Models with Patient Workload and Nurse *Preferences*", *Management 2012*. 2(5): 149-160 -DOI: 10.5923/j.hrm.20120205.03.
[27] K. Rowland, "Nurse scheduling with tabu search and strategic oscillation", European Journal of Operational Research 106 393–407, 1998.
[28] M.J. Brusco, L.W. Jacobs. "Cost analysis of alternative formulations for personnel scheduling in continuously operating organizations" European Journal of Operational Research 86 249–261, 1995.
[29] M. Warner," Scheduling Nursing Personnel According to Nurse Preference: a Mathematical Programming Approach.". *OR*.24: 842-856. 1976.
[30] M. V. Chiaramonte, *Competitive nurse rostering and rerostering*. Arizone State University, 2008.
[31] P. Ross, "Hyper-heuristics' E.K. Burke, G. Kendall, eds. Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies. Chapter 16, Springer, 2005.

Geetha B. was born in Melaka, Malaysia. She is an Assistant Professor at the University of Nottingham, Malaysia Campus. She is a member of the Automated Scheduling and Planning research group in the School of Computer Science at the University of Nottingham. She is currently pursuing her PhD studies here focussing on Nurse Scheduling. Her main research area include nurse scheduling, domain transformation, information granulation, heuristic, IP.ILP, matrix exploration.

Andrzej B. is Professor in the School of Computer Science at the University of Nottingham. He served as President of the European Council for Modelling and Simulation (ECMS) during 2002-2006 and 2010-2012. He is Associate Editor of the IEEE Transactions on Systems Man and Cybernetics and Associate Editor of the Information Sciences. His research involves investigation into Granular Computing, human-centred information processing as a methodological approach to solving large-scale data mining and system complexity problems.

Rong Q. is a Lecturer in the School of Computer Science at the University of Nottingham. She gained her PhD in Computer Science from the University of Nottingham in 2002. Her main research areas include meta-heuristics, constraint programming, IP/ILP, case based reasoning methodologies and knowledge discovery techniques on scheduling, especially educational timetabling, healthcare personnel scheduling, network routing problems and graph colouring. In total she has more than 30 papers published or to appear at international journals and peer-reviewed international conferences. Dr Qu is also a guest editor for special issues at the journal of Memetic Computing and the Journal of Scheduling, and the program chair of several workshops and an IEEE symposium.

Instance	Best known cost	BR [8] Cost	BR [8] time(s)	MEH [9] Cost	MEH [9] time(s)	BUR 09b (SS2) Cost	BUR 09b (SS2) time(s)	BUR 09b (MEH) Cost	BUR 09b (MEH) time(s)	Our approach Cost	Our approach time(s)
Musa	**175**	175	<0.1	175	39					175	1
GPost	**5**	5	2	8	234	9	4305	915	605	5	5
GPost-B	**3**	3	29			5	3955	789	475	3	10
Ozkarahan	**0**	0	<0.1	0	1					0	<0.1
Millar-2Shift-Data1	**0**	0	<0.1	0	1	0	910			0	1
Millar-2Shift-Data1.1	**0**	0	<0.1			0	20			0	2
Azaiez	**0**	0	0.3	0	233					0	2
WHPP	**5**	5	17.6							5	10
Valouxis-1	**20**	80	909.6	160	3780	100	4000			20	40
Ikegami-2Shift-Data1	**0**	0	59.7	0	671					0	40
Ikegami-3Shift-Data1	**2**	2	59.8	63	2671					2	55
Ikegami-3Shift-Data1.1	**3**	4	995.2							3	85
Ikegami-3Shift-Data1.2	**3**	5	5411.9							3	95
ORTEC01	**270**	270	69.3			365	3400	535	7580	**130**	**85**
ORTEC02	**270**	270	105.1							**130**	**99**
QMC-1	**13**	13	57.6			20	4435	39	3160	13	60
QMC-2	**29**	29	1.9							29	2

TABLE 2: Results for Greedy Benchmark Instances

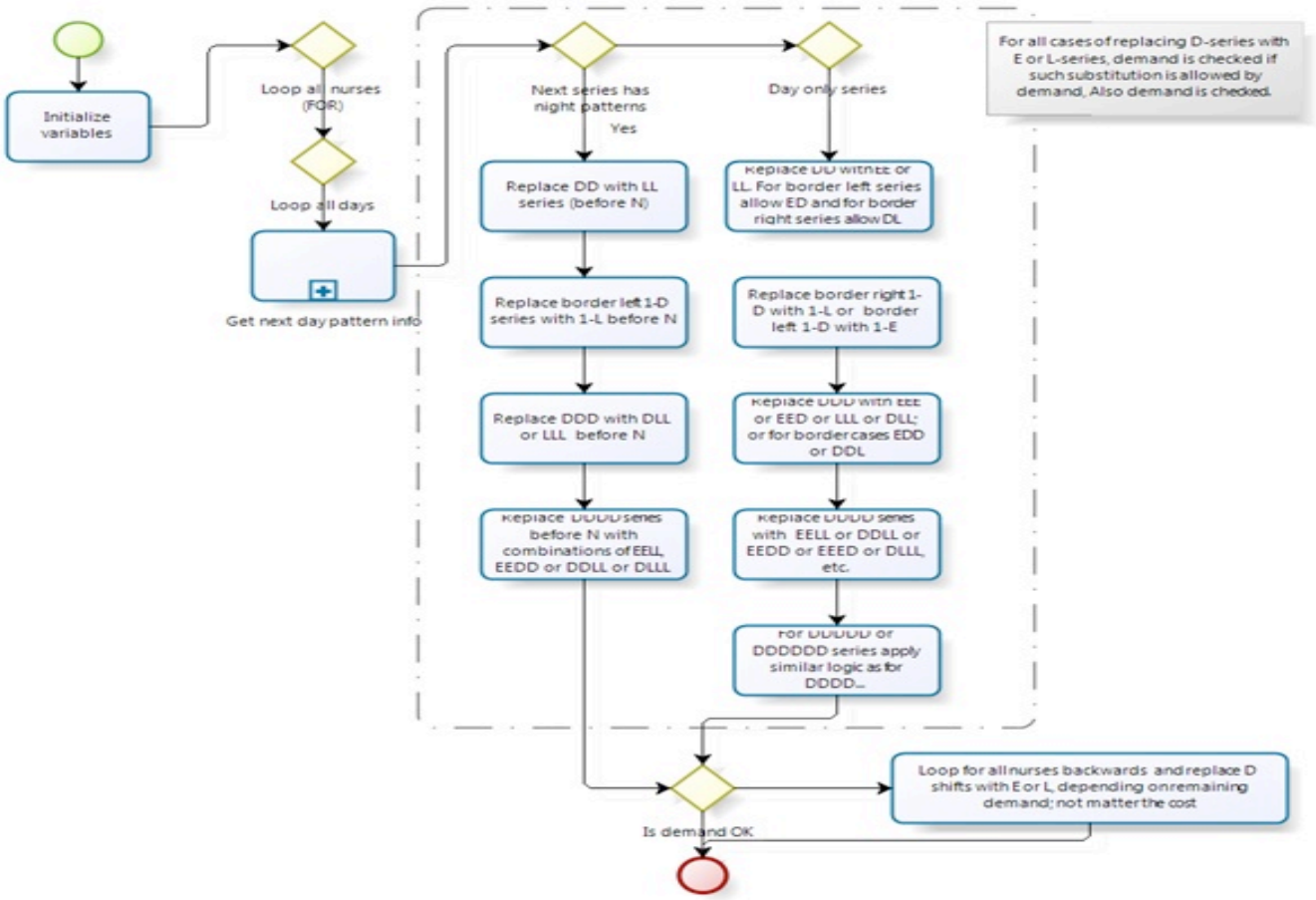

Figure 6: Process of converting DNR to edlNR