Investigation into an Evolutionary Algorithm Hyper-Heuristic for the Nurse Rostering Problem

Christopher Rae
Nelishia Pillay
School of Mathematics, Statistics and Computer Science
+27 33 2605644
stingrae789@gmail.com; pillayn32@ukzn.ac.za

Abstract: Nurse rostering is a well researched domain with various methods successfully applied to solving this problem including constraint programming, integer programming, simulated annealing, tabu search and genetic algorithms. The effectiveness of hyper-heuristics in solving combinatorial optimization problems of this nature has been illustrated in its application to educational timetabling and packing problems amongst others. However, there has not been much research into the use of hyper-heuristics in solving the nurse rostering problem. In particular, multi-point search methods, namely population based hyper-heuristics such as evolutionary algorithm hyper-heuristics, have not been investigated for this domain. The research presented in this paper forms part of a larger initiative aimed at researching the use of evolutionary algorithm perturbative hyper-heuristics in solving the nurse rostering problem. The paper reports on the application and evaluation of an evolutionary algorithm selection perturbative hyper-heuristic (EA-SPHH) in solving the benchmark data set used for the first international nurse rostering competition.

Keywords: hyper-heuristics, nurse rostering, evolutionary algorithm hyper-heuristic

1. Introduction

The nurse rostering problem (NRP) is a two-dimensional timetabling problem that deals with the assignment of nursing staff to shifts across a scheduling period subject to certain constraints (Burke et al. 2004). The constraints are divided into hard constraints, constraints which must be fulfilled, and soft constraints, i.e. those that are desirable. These constraints are set by the hospital's contracts for the staff and their own personal requests. This problem has been proven to be NP-hard.

Hyper-heuristics is a problem solving approach, that aims to have a higher level of generality than traditional optimization methods by operating on a heuristic space rather than a solution space (Burke et al. 2013). Hyper-heuristics select or generate low-level heuristics. The low-level heuristics can be constructive or perturbative. Constructive heuristics are used to construct a solution while perturbative heuristics improve a candidate solution. This study focuses on selection perturbative hyper-heuristics. These hyper-heuristics are categorized as single-point or multi-point.

Previous research on hyper-heuristics for the domain of nurse rostering have essentially been single point selection perturbative hyper-heuristics. Cowling et al. (2002) used a choice function selection perturbative hyper-heuristic to solve 52 instances of the nurse rostering problem for a UK hospital. Burke et al. (2003) employed a selection perturbative hyper-heuristic with a tabu list memory to the same problem set with an improvement in performance. Bai et al. (2010) apply a selection perturbative hyper-heuristic hybridized with a genetic algorithm to solve the same set of problems which produced further improvement in results. Heuristics are chosen based on dynamic performance and acceptance ratios and simulated annealing is used for move acceptance. The hyper-heuristics implemented by Bilgin et al. (2009) to solve the Belgian nurse rostering problem used random heuristic selection and simulated annealing for move acceptance. In later work Bilgin et al. (2010) extend this work by hybridizing a selection perturbative hyper-heuristic and a greedy shuffle approach to solve problems from the first international nurse rostering competition.

Previous research has examined the use of single point selection perturbative hyper-heuristics in solving the nurse rostering problem. Given the success of multi-point hyper-heuristics in other domains such as educational timetable (Burke et al. 2013), this study investigates the use of a multi-point selection perturbative hyper-heuristic, namely an evolutionary algorithm hyper-heuristic, in solving this problem. This study extends the work presented in Rae et al. (2012).

2. Evolutionary Algorithm Selection Perturbative Hyper-Heuristic (EA-SPHH)

The EA-SPHH employs the generational evolutionary algorithm depicted in Figure 1.

Create initial population
Repeat
    Evaluate individuals in the population by applying each heuristic in the individual to the current solution
    Set best individual's solution as the current roster
    Select parents using tournament selection
    Apply genetic operators to selected parents and evaluate individuals in population
Until a maximum number of generations has been reached or the solution has converged

Figure 1. Generational evolutionary algorithm hyper-heuristic

An initial population is created, this population is then refined iteratively following a process of evaluation, selection of parents and use of genetic operators to create offspring which form the next generation's population. An individual in the population is represented by a string with each character representing a low-level perturbative heuristic. A number of heuristics were tested and developed based on literature and it was decided to use a set of 13 swap operators. In addition to this a "blank" move is included as a means of introducing entropy into the system. Each element of the string is randomly chosen until a string within a maximum and minimum length is created.

The fitness measure of each element of the population is determined by using the individual to improve a nurse roster created by random allocation of shifts to nurses. The fitness of the individual is the sum of the hard and soft constraints violated by the constructed roster. For each generation the best individual's roster replaces the initial roster. This is a form of shared memory as we are giving information to all the individuals in the population. The system and chosen heuristics avoid incurring hard constraint penalties. This is done by only having as many shifts as defined by the cover requirement and only allowing nurses to have single shift days in the roster representation. Tournament selection is used to choose parents for regeneration which involves application of mutation and crossover operators. This selection method returns the fittest individual of a randomly selected number of individuals.

In each generation the population is created by applying the mutation and crossover operators on the selected parents. The mutation operator selects a random position in a parent string and changes the heuristic at that point to a random heuristic from the given set to produce an offspring. The crossover operator randomly selects two points in each of the parents. The substrings at these points are swapped to create two offspring. The fitter offspring is returned as the result of the operation.

Multithreading was introduced into the algorithm in order to improve runtimes and simulations were run on a multicore machine using 8 processors of the available 128 cores per run. Table 1 lists the parameters values used by the evolutionary algorithm which were determined empirically by performing trial runs.

Table 1. Parameter values

Parameter	***Value***
Population Size	100
Initial Chromosome Length	10-25×Cover Size
Tournament size	5
Crossover percentage	70%
Mutation percentage	30%
No. of Runs	10
Maximum no. of Generations	20

4. Results and Discussion

This section discusses the performance of the of the EA-SPHH in solving the benchmark problems of the first international timetabling competition. This included 30 sprint instances (10 early, 10 late and 10 hidden) 15 medium instances (5 early, 5 late and 5 hidden) and 15 long instances (5 early, 5 late and 5 hidden. EA-SPHH was able to produce optimal results for 20 of the sprint instances. For the remaining sprint problem instances the EA-SPHH produced solutions that differed by 1 for 4 instances, 2 for 2 instances, 3 for 1 instance, 5 for 2 instances and 6 for 1 instance. The performance of the EA-SPHH was not as good on the medium and long instance, producing optimal solutions for 7 of the medium instances and 4 of the long instances. The difference from the optimal ranged from a minimum of 1 to a maximum of 36. Forty six of 60 instances were either solved to optimality or within 5 soft constraint violations from the best known solution. The performance of the hyper-heuristic does not compare with the state of the art approaches, however this was not expected as this is a first attempt at employing a multi-point search in solving this problem. The performance of the EA-SPHH is better than the selection perturbative hyper-heuristic hybridized with a greedy shuffle implemented by Bilgin et al. (2010) on the sprint instances and is comparative on the medium and some of the long instances, with the hybrid implemented by Bilgin et al. performing better on the late variant of the long instances. The EA-SPHH was also found to perform much better than the harmony search employed by Awadallah et al. (2011) in solving these problems. EA-SPHH was able to match the solution found by Bilgin et al. (2010) on long_hidden02, the solution found by Burke and Curtois (2010) for the sprint_late04 instance and the solution found by Nonobe (2010) for the sprint_hidden01 instance. These results were previously the best known results. Future research will investigate means of improving the performance of the EA-SPHH. It is anticipated that one of the reasons for the poor performance on medium and late instances is the set of low-level heuristics used. The EA-SPHH essentially uses swap heuristics. The heuristic set does not include ruin and recreate or mutational and crossover heuristics as used in some of the previous work on hyper-heuristics. Future research will investigate methods for investigating the most effective set of heuristics to be used by the EA-SPHH. Furthermore, too big a set of low-level heuristics will lead to a greater number of combinations resulting in a larger search space to explore and possibly poor success rates. In this study a total of 14 low-level heuristics were used.

5. Conclusion and Future Work

The paper is an initial attempt at employing a multi-point selection perturbative hyper-heuristic to solve the nurse rostering problem. An evolutionary algorithm selection perturbative hyper-heuristic was implemented and evaluated on the benchmark set of problems used for the first international nurse rostering competition. While the hyper-heuristic produced good results for the sprint set of problems it did not perform as well on the medium and long instances, particularly on the late problem type. It is hypothesized that this could possibly be attributed to the set of low-level heuristics used and future work will investigate methods for determining the set of low-level heuristics to use. Future work will also investigate the use of a generative perturbative hyper-heuristic to evolve low-level perturbative heuristics and a hybrid hyper-heuristic combining both selection and generation perturbative hyper-heuristics to solve this problem.

Acknowledgements. This work is based on the research supported in part by the National Research Foundation of South Africa for the Grant CSUR13091742778. Any opinion, finding and conclusion or recommendation expressed in this material is that of the author(s) and the NRF does not accept any liability in this regard.

7. References

Awadallah, M. A., Khader, A. T., Al-Betar, M. A., & Bolaji, A. L. (2011). Nurse Scheduling Using Harmony Search. In proceedings of the Sixth International Conference on Bio-Inspired Computing: Theories and Applications, 58–63.

Bai, R., Burke, E. K., Kendall, G., Li, J. & McCollum, B. (2010) A Hybrid Evolutionary Approach to the Nurse Rostering Problem. *IEEE Transactions on Evolutionary Computation*, 14(4), 580–590.

Burke, E. K., & Curtois, T. (2010) An Ejection Chain Method and a Branch and Price Algorithm Applied To The Instances Of The First International Nurse Rostering Competition. Technical Report, School of Computer Science, University of Nottingham.

Bilgin, B., De Causmaecker, P. & Vanden Berghe, G. (2009) A Hyper-Heuristic Approach to Belgian Nurse Rostering. In proceedings of the Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2009), pp. 683-689.

Bilgin, B., Demeester, P., Misir, M., Vancroonenburg, W., Vanden Berghe, G. & Wauters, T. (2010) A Hyper-Heuristic Combined with a Greedy Shuffle Approach to the Nurse Rostering Problem. Online. https://www.kuleuven-kulak.be/~u0041139/nrpcompetition/abstracts/ l3.pdf.

Burke, E. K., De Causmaecker, P., Vanden Berghe, G. & Van Landeghem, H. (2004) The State of the Art of Nurse Rostering. *Journal of Scheduling*, 7(6), 441–499.

Burke, E. & Soubeiga, E. (2003) Scheduling Nurses Using a Tabu-Search Hyper-Heuristic. In Proceedings of First Multidisciplinary International Scheduling Conference: Theory and Applications, pp. 197-218.

Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E. & Qu, R. (2013) Hyper-Heuristics: A Survey of the State of the Art. *Journal of the Operational Research Society*, 1-30.

Cowling, P., Kendall, G., Soubeiga, E. (2002) Hyperheuristics : A Robust Optimisation Method Applied to Nurse Scheduling. *Parallel Problem Solving from Nature — PPSN VII Lecture Notes in Computer Science*, 2439, 851–860.

Haspelagh, S., De Causmaecker, P., Stolevik, M., Schaerf, A. (2012) The First International Competition on Nurse Rostering 2010. *Annals of Operations Research*. DOI: 10.1007/s10479-012-1062-0.

Nonobe, K. (2010) INRC2010 : An Approach Using a General Constraint Optimization Solver. https://www.kuleuven-kulak.be/~u0041139/nrpcompetition/abstracts/m3.pdf.

Rae, C., Pillay, N. (2012) A Preliminary Study into the Use of an Evolutionary Algorithm Hyper-Heuristic to Solve the Nurse Rostering Problem, in Proceedings of the 4th World Congress on Nature and Biologically Inspired Computing (NaBIC 2012), pp. 156-161.