Variable Neighbourhood Search: A Case Study for a Highly-Constrained Workforce Scheduling Problem

Kenneth N. Reid, Jingpeng Li
Division of Computing Science and Mathematics
University of Stirling
Stirling FK9 4LA, UK

Jerry Swan
Department of Computing Science
University of York
York YO10 5GH, UK

Alistair McCormick, Gilbert Owusu
BT Research & Innovation
PP MLB3/PP12, Orion Building
Adastral Park, Martlesham Heath, IP5 3RE, UK

***Abstract*— This paper describes a Variable Neighbourhood Search (VNS) combined with Metropolis-Hastings acceptance to tackle a highly constrained workforce scheduling problem typical of field service operations (FSO) companies. A refined greedy algorithm is firstly designed to create an initial solution which meets all hard constraints and satisfies some of the soft constraints. The VNS is then used to swap out less promising combinations, continually moving towards a optimal solution until meeting finishing requirements, which are either a satisfactory mean fitness set as a parameter, or a time allowance of one hour. The results of this approach are promising when compared to the stand-alone greedy algorithm.**

Keywords—Variable Neighbourhood Search, Personnel Scheduling, Engineer Rostering, Metaheuristic

I. INTRODUCTION

When there are tens or hundreds of employees, with hundreds or thousands of shifts requiring employee allocations, an exponential number of possible solutions emerge. In order to decide which of the numerous potential and acceptable solutions are closer to optimal; definitions of preferable solution attributes are required. These criteria are known as Hard Constraints (HCs) and Soft Constraints (SCs) [1]. HCs are often simpler requirements which in this case are Boolean in nature; where a fail will cause the solution to become unusable due to contractual obligations, legal obligations or other highly important constraints. In the case of employee scheduling, an example would be breaking employment laws such as the legal number of hours an employee can work per week. SCs are the measurements of multiple criteria which describe a usable solution and its attributes on a percentage scale of potential fitness. An example of a SC in this problem is aiming to satisfy the workforce by ensuring as many employees have consecutive rest days per week as is possible, as opposed to rest days with working day in between them. While there are solutions which exist to deal with subsets of the constraints presented, this case study provides a solution to the unique problems large number of specific constraints.

Personnel scheduling problems have been widely studied in recent decades, and there have been calls for further research into multiple decision variables that are integrated into the solution engine [2]. Demand forecasting being one such suggested topic for further study, which we have implemented in this case study. Multiple fields have benefited from the use of metaheuristics to solve large complex rostering scheduling problems, including nurse rostering [3,4,5], truck scheduling [6] and exam timetabling [7,8]. It goes to follow that a similarly highly constrained search optimization problem in engineer rostering can benefit from similar techniques.

There have been multiple solutions provided to similar problems, using techniques such as tabu search [9], simulated annealing [10], as well as different applications of variable neighbourhood search [11]. These examples of metaheuristics have a proven track record with less constrained and smaller search spaces but the use of a greedy algorithm to create an initial solution space which is further optimized by Variable Neighbourhood Search (VNS), yet monitored with Metropolis-Hastings acceptance shows promise with the highly constrained search space unique to this problem.

A further advantage of this approach is the combination of benefits from both algorithms: A greedy algorithm creating the initial solution set is limited by the inability to move around the solution space to a closer to optimal result, often because of a requirement to cross an infeasible area. The further use of VNS allows a broader search, furthering the potential solution space by use of three techniques: Swapping shifts between two separate employees on an individual day (e.g. employee A's day shift and employee B's night shift); moving an employee to work a separate day entirely (e.g. moving Employee A from a Monday shift to a Saturday shift). Finally, Metropolis-Hastings acceptance is used to decrease the likelihood of the more disruptive approach of swapping of days being used as opposed to swapping shifts, which is less disruptive. This is important to prevent the accidental corruption of a good solution into a worsened one. Metropolis-Hastings acceptance is used to decrease the likelihood of larger neighbourhoods being swapped, later in run time.

Similar problems including telecommunications companies scheduling employees to meet a variety of tasks have been tackled in the past, e.g. [12]. These types of tasks have been described as technician and task scheduling problems (TTSP) in survey papers [13].

This paper is organized as follows: First, the problem description is defined in section II, where the key characteristics of the problem are described, including a definition of each hard

and soft constraint. Section III defines the problem formally with an Integer Programming (IP) model (for more details see [14]). Section IV includes the VNS controlled and monitored by Metropolis-Hastings acceptance, making use of a solution provided by a greedy algorithm in order to achieve a closer to optimal solution is also further detailed in this section. The results of testing are presented in Section V. The final section of this paper is the conclusion and discussion on possible future research directions.

II. Problem Description

The engineer rostering problem is typical of large industrial companies which operate large scale field engineering workforce to service their customers. The algorithm is based on scheduling one of many sets of engineers, ranging up to 150 employees per run. This assigns any number of two types of shifts, day and late, within a specified scheduling period, which itself has no upper limit in length, but for testing purposes three months of rostering is generated with the engine.

The problem has the following key characteristics:

1. UK Employment Law must be adhered to, including maximum working hours allowed per week, number of rest hours in between shifts, etc.

2. While a solution cannot fail when employees don't have two consecutive rest days within a week, it is highly important. Similarly shift types and less frequent day types (e.g. uncommon Saturday shifts) must be spread through the roster where possible. These and other constraints which rate the quality of a solution are described within the soft constraints.

3. Engineers have largely varied skillsets, both in skills available as well as each engineer has a skill preference within which the higher preference skills should be selected before lower preference skills. Demand may differ largely on a day to day basis, as well as week to week. Demand is typically based on demand predictions created historical work patterns. The objective is to assign as many engineers to shifts where they are most appropriate for the task as opposed to working a shift only to meet contractual obligations, i.e. it's better to meet soft constraints and hard constraints rather than just hard constraints.

4. We assume that each employee has at least one of each shift type in the roster.

5. Demand is met in order of priority, which is handled by processing data as it is input to the engine. Similarly, each employee has a set of skills, which are ordered into skill preference. Employees with a higher skill preference for a specific skill required on a shift must be allocated before employees with lower skill preference, or undesirable skills.

This paper describes the module known as "RosterGen" in Fig. 1, with demand and employee data as an input, and the near optimal roster as an output. The algorithm described in this paper is the means by which the engine produces a roster. The other sections are simply user interface, visualization and database models, however it is useful to see the construction of the algorithm within this engine.

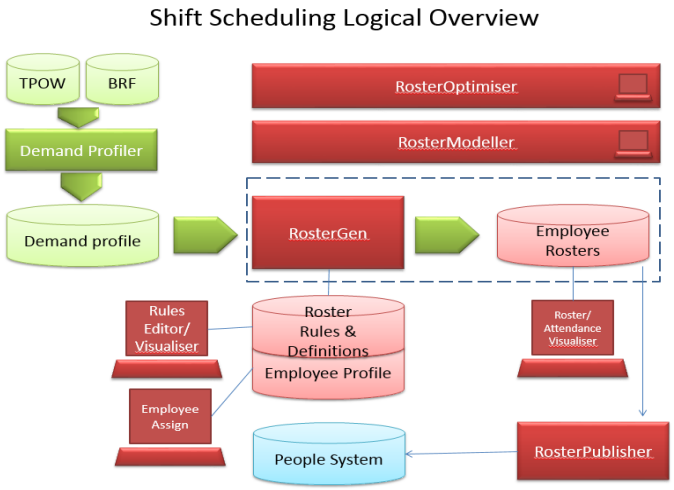

Fig. 1 A visualized logical overview of the shift scheduling solution.

A. Hard Constraints

HC1: Every employee must have 0 or 1 shift per day.

HC2: Every employee has a maximum number of day types (Monday - Sunday) in the scheduling period. A possible maximum is no restriction (-1), none of that day type allowed (0) or an enforced upper threshold (> 0).

HC3: Every employee has a minimum number of day types (Monday - Sunday) in the scheduling period. A possible minimum is no restriction (-1) or an integer which represents the enforced lower threshold (> 0).

HC4: Every employee has a maximum number of DAY shift type in the scheduling period. A possible maximum is no restriction (-1), no DAY shift type allowed (0) or an enforced upper threshold (> 0).

HC5: Every employee has a minimum number of DAY shift type in the scheduling period. A possible minimum is no restriction (-1), or an integer which represents the enforced lower threshold (> 0).

HC6: Every employee has a maximum number of LATE shift type in the scheduling period. A possible maximum is no restriction (-1), no LATE shift type allowed (0) or an enforced upper threshold (> 0).

HC7: Every employee has a minimum number of LATE shift type in the scheduling period. A possible minimum is no restriction (-1), or an integer which represents the enforced lower threshold (> 0).

HC8: Every employee with the A_1 day shift pattern type must be allocated to A_1 shifts every week.

HC9: Every employee with the A_2 day shift pattern type must be allocated to A_2 shifts every week.

HC10: Every employee with the A_1/A_2 day shift pattern type must be allocated to A_1 shifts every second week, and A_2 shifts every other week.

B. Soft Constraints

SC1: Demand requirements of every shift on every day.

SC2: Two consecutive rest days for every employee on every week.

SC3: Every employee's least common shift type (DAY or LATE shift) must be spread as evenly as possible over the scheduling period to allow a high as possible mean number of days between the least common shift type allocations.

SC4: Every employee's less common day types (Monday – Sunday) must be spread as evenly as possible over the scheduling period to allow a high as possible mean number of weeks between the less common day type allocations. Not used if employee is scheduled for one, less than one, or the maximum number of day type(s), as ideal intermittency already achieved.

SC5: The number of employees allocated to each day type (Monday – Sunday) must be spread as evenly as possible over the scheduling period.

III. An Integer Programming Model

Following an integer programming model, similar to that seen elsewhere in the literature [15,16], the following model is used to demonstrate the problem this paper tackles. Slack and surplus variables can be introduced to the soft constraints, and the objectives are to minimize the values of individual variables. We formulate the entire problem associated with a variable scheduling period as the following IP model, which can be altered to adapt to other problems with different constraints. This approach can be used for any length of scheduling period.

TABLE I. Integer Programming Model Parameters

Parameters	
I	Set of employees available
$I_t \mid t \in \{1,2,3\}$	Subset of employees that work A_1, A_2 and A_1/A_2 shift patterns respectively. $I = I_1 + I_2 + I_3$
J	Set of days representing days in a week (1-7, i.e. Monday – Sunday)
W	Set of all weeks in the scheduling period. $W = W_1 + W_2$, where W_1 represents the weeks with odd indices and W_2 the weeks with even indices
K	Set of shift types = {1 (day shift), 2 (late shift)}
Z	Set of skills, in order of preference when used in reference to an employee, in order of demand priority when used in relation to days or weeks
D_{zkjw}	Demand requirement for a skill z on a shift type k, on a day j, in a week w
α_{ij}	The maximum number of day types over the scheduling period for an employee
β_{ij}	The maximum number of DAY shift types over the scheduling period for an employee
γ_{ij}	The maximum number of LATE shift types over the scheduling period for an employee
ζ_{ij}	The minimum number of day types allowed over the scheduling period for an employee.
η_{ij}	The minimum number of DAY shift types over the scheduling period for an employee
μ_{ij}	The minimum number of LATE shift types over the scheduling period for an employee
l_{ik}	The number of occurrences of shift type k for each employee i over the scheduling period. $l_i \leq 5\|W\|$
k_i	The least common shift type for employee i, $k_i \in K$. $k_i = 1$ if $l_{i1} < l_{i2}$, 2 otherwise
R	Set of the day index for each occurrence of k_i for employee i over the scheduling period. $= \{r_1, r_2, \ldots\}$, $\|R\| = l_{ik}$
o_i	Ideal intermittence between the i-th employee's k_i, $o_i = \text{int}(7\|W\| / l_{ik_j})$
p_{ij}	Number of shifts worked for employee i on day j. $p_{ij} = \sum_{k \in K} \sum_{w \in W} x_{ijkw}$, and $p_{ij} > 1$
m_{ik_i}	The number of occurrences of the least common shift type k_i for each employee i on day j. $m_{ik_i} < \|W\|$
q_i	Ideal day intermittence for employee i, $q_i = \left\| \|W\| / m_{ik_i} \right\|$, $q_i \in \mathrm{N}/\{0\}$
t_{jw}	Number of employees allocated per j across W, $t_{jw} = \sum_{i \in I} \sum_{k \in K} x_{ijkw}, \ \forall j \in J, w \in W$
u_{jw}	Daily allocated employee spread fitness score per j across W, $u_{jw} \in \{0,1\}$
v_j	Mean allocated employees per j, $v_j = \left(\sum_{i \in I} \sum_{k \in K} \sum_{w \in W} x_{ijkw} \right) \Big/ \|W\|, \ \forall j \in J$
$v_j \pm \delta$	Acceptable upper / lower bounds of t_j per j across W. $\delta \geq 0$, δ is an input to the algorithm, which depends on business preferences at the time

Fig. 2 Integer Programming Model Parameters

Decision variable x_{ijkw} is 1 if an employee i is assigned shift type k for day j in week w, 0 otherwise, for all employees on day shifts on all weeks on all shifts, defined as:

$$x_{ijkw} = 0 \, or \, 1, \ \forall i \in I, j \in J, w \in W, k \in K \quad (1)$$

Slack / surplus variables are minimized and are used as the positive or negative deviations from individual goals, defined as:

$$s^1_{zjkw} \geq 0, s^2_{zjkw} \geq 0, \forall z \in Z, \ j \in J, k \in K, w \in W \quad (2)$$

$$s^3_{ijw} \geq 0, s^4_{ijw} \geq 0, \forall i \in I, \ j \in J, w \in W \quad (3)$$

$$s^5_{ti} \geq 0, \ \forall r_t \in R, i \in I \quad (4)$$

$$s^6_i \geq 0, \ \forall i \in I \quad (5)$$

$$s^7_j \geq 0, \ \forall j \in J \quad (6)$$

Subject to:

$$\text{HC1} \ \sum_{k \in K} x_{ijkw} \leq 1, \ \forall i \in I, \ j \in J, w \in W \quad (7)$$

$$\text{HC2} \ \sum_{k \in K} \sum_{w \in W} x_{ijkw} \leq \alpha_{ij}, \ \forall i \in I, \ j \in J \quad (8)$$

$$\text{HC3} \ \sum_{k \in K} \sum_{w \in W} x_{ijkw} \geq \zeta_{ij}, \ \forall i \in I, \ j \in J \quad (9)$$

$$\text{HC4} \ \sum_{w \in W} x_{ij1w} \leq \beta_{ij}, \ \forall i \in I, \ j \in J \quad (10)$$

$$\text{HC5} \ \sum_{w \in W} x_{ij1w} \geq \eta_{ij}, \ \forall i \in I, \ j \in J \quad (11)$$

$$\text{HC6} \ \sum_{w \in W} x_{ij2w} \leq \gamma_{ij}, \ \forall i \in I, \ j \in J \quad (12)$$

$$\text{HC7}\ \sum_{w\in W} x_{ij2w} \geq \mu_{ij},\ \forall i \in I, j \in J \quad (13)$$

$$\text{HC8}\ \sum_{j\in J}\sum_{k\in K} x_{ijkw} = A_1,\ \forall i \in I_1, w \in W \quad (14)$$

$$\text{HC9}\ \sum_{j\in J}\sum_{k\in K} x_{ijkw} = A_2,\ \forall i \in I_2, w \in W \quad (15)$$

$$\text{HC10}\ \sum_{j\in J}\sum_{k\in K} x_{ijkw} = A_1,\ \forall i \in I_3, w \in W_1 \quad (16)$$

$$\sum_{j\in J}\sum_{k\in K} x_{ijkw} = A_2,\ \forall i \in I_3, w \in W_2 \quad (17)$$

$$\text{SC1}\ \sum_{i\in I} x_{ijkw} + s^1_{zjkw} - s^2_{zjkw} = d_{zjkw},$$
$$\forall z \in Z, j \in J, k \in K, w \in W \quad (18)$$

$$\text{SC2}\ \sum_{k\in K}[x_{ijkw} + x_{i(j+1)kw}] + s^3_{ijw} - s^4_{ijw} = 0,$$
$$\forall i \in I, j \in J, w \in W \quad (19)$$

$$\text{SC3}\ r_{t+1} - r_t + s^5_{ti} \leq o_j, \forall r_t \in R, i \in I \quad (20)$$

$$\text{SC4}\ [(\sum_{p=1}^{p-\varepsilon} w_{(p+\varepsilon)_{ij}} w_{p_{ij}}) + 1] / p_{ij} + s^6_i \leq q_i,\ \forall i \in I$$
where $p+\epsilon$ is next p_{ij} after current p_{ij} across W (21)

$$\text{SC5}\ (\sum_{w\in W} u_{jw}) / |W| + s^7_j = 1,\ \forall j \in J$$
$$u_{jw} = 1 if\ t_{jw} \in [v_j - \delta, v_j + \delta], else\ u_{jw} = 0 \quad (22)$$

IV. A Metropolis-Hastings acceptance Based Variable Neighbourhood Search

A. Greedy Algorithm

The first step of this solution is to create a set of neighbourhoods which satisfy the hard constraints as defined in Section II. This process is demonstrated in Fig 3. The counter *c* represents the current number of shifts still required to meet all employee's contractual obligations.

```
GreedyAlgorithm(Employees e in E, Demand d){
        Define a set of Weeks w in W;
        Define shifts k in K;
        Define skills s in S;
        Define c;
        Loop while c > 0{
                Get random k from W;
                If d not met on k{
                        Find highest priority d
                        in k;
                        Find available e with
                        highest skill preference
                        k for d;
                        Allocate e to k;
                        c--;
                }
        }
}
```

Fig. 3 *Pseudo-code of the Greedy Algorithm*

B. Variable Neighbourhood Search

The VNS requires a solution, comprised of neighbourhoods, as a parameter. A benefit of using the VNS is that, while the Greedy Algorithm limits the scope of local optima, the VNS allows some movement around the solution space, providing a higher chance of reaching the global optimum.

The solution the VNS receives is a set of weeks with employees allocated throughout. The VNS works in three steps as follows.

Firstly, by looking at a random day within the solution which has at least two employees on different shifts (day shift and late). Then a hard constraint check is run, ensuring that swapping these two employees is acceptable. If they can safely be swapped, the algorithm then analyses the current solution in terms of fitness; how well the soft constraints are met. The swap occurs, and a second fitness test is run. If the fitness has worsened, the swap is reverted, otherwise the change stays.

The percentage chance of the next phase occurring is dependent on Metropolis-Hastings acceptance, which is described in section C. In this phase, a more disruptive change occurs: instead of swapping employees on a single day, an employee is moved shifts from one day to another in the same week. This is dependent on whether the hard constraints allow this swap (e.g. if an employee cannot contractually work Sundays, they will never be given a Sunday). Similarly to before, this change will occur, but only if fitness improves will the change remain as part of the solution.

The final phase of the VNS is the least likely to occur as it is the most destructive, and is also controlled by Metropolis-Hastings acceptance, preventing this from occurring as frequently towards the end of runtime. In this phase a single employee has a week of allocations removed. The employee is then given a new set of shifts which meet Hard Constraints, and to an extent Soft Constraints. Regardless of improvement of fitness or not, this change remains. This is intentional, in order to leave the current neighbourhood and potentially reach higher fitness levels.

C. Metropolis-Hastings acceptance

In this instance the solution utilizes the probabilistic acceptance criterion of Metropolis-Hastings acceptance, similar to exponential Monte-Carlo or Metropolis-Hastings acceptance [17], and embedding it within the VNS.

Unlike in more traditional methods, in this approach Metropolis-Hastings acceptance is used to attempt to find a global maximum. Specifically, this method of controlling the frequency of an event occurring over time is used to reduce the likelihood of disruption the closer the algorithm is to ending. The VNS should produce a better solution as time moves forward, but if the VNS is not controlled by Metropolis-Hastings acceptance there is increased chance of losing a more highly rated fitness solution.

The only parameter used during Metropolis-Hastings acceptance is based on length of runtime. The goal of the Metropolis-Hastings acceptance is to reduce the likelihood of unnecessarily altering a good solution later in runtime, as such the chance of a drastic change at the beginning is 10%, and this reduces by 50% of the current likelihood every 10% of runtime (i.e. 10% at start, then 5%, then 2.5%, etc.) This means it's still possible a change will happen later in runtime but becomes much less likely, thus increasing likelihood of a good solution.

V. Results

Whether a solution can be accepted as feasible or infeasible is determined by the hard constraints being achieved, which ensure that legal and contractual requirements are met. The quality of the feasible solutions is measured by how well the soft constraints are met. Solutions produced by this algorithm always meet hard constraints, and meet soft constraints to some extent. For this paper, we assume the soft constraints are of equal priority, however it should be noted that these can be modified before run time so certain soft constraints take a higher precedence and the final fitness results are weighted as such.

Fitness results are obtained from fitness functions, which each return a value pertaining to the soft constraint they represent. The fitness results are then averaged to return an overall fitness value for the current solution.

The results in this section were tested on an oracle enterprise Linux 6 OS server with a 12 core Xeon CPU for 1 hour.

The tests used real employee data and real demand forecast data, however it should be stated that the number of shift types the employees work was increased as the data provided was for an area where engineers worked a very small number of late shifts per quarter. There were 50 tests conducted, and the results are shown in Fig 4. These results show the mean fitness of the solution after being processed by the greedy algorithm compared to running variable neighbourhood search processes for one hour. These values are calculated by taking the mean of all fitness values carried out from the soft constraints (see section III). In practice the soft constraints can be weighted so the engine can be used to produce solutions which cater to specific intrigues, for example increasing the weighting of the demand soft constraint being met, or employees longer rest break periods.

The mean of all greedy algorithm results below is 0.681309278934374 and the mean of all VNS results is 0.784683675825638, which indicates that over fifty tests running at 1 hour each, there is an average of 10.3% increase in fitness using the VNS algorithm after the greedy algorithm sets up an initial solution.

Metropolis-Hastings acceptance is the technique we used in order to prevent a closer to optimal solution from being disrupted later in the runtime. The runtime parameters are e = 100% chance of acceptance at the beginning, with an e * 0.99 reduced chance of disruption per iteration. The results of each run are as expected. While the greedy algorithm can find a suitable solution, the VNS improves upon each solution by iteratively meeting more soft constraints.

The coefficient of determination (R^2) is shown in Fig 6. This statistic gives useful information about the goodness of fit. An R^2 value of 1 would indicate the regression line perfectly fits the model. In this instance the R^2 value is calculated at 0.83, this is a good indicator of positive correlation between the results of the VNS algorithm and the Greedy Algorithm.

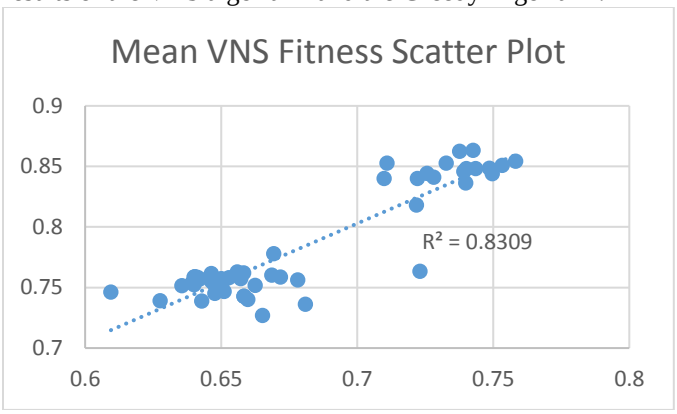

Fig. 4

A question of efficiency can be raised at this point: after using a server with comparatively substantial resources for 1 hour per test, what is the most efficient time to leave this process running for? This is a point for future research.

VI. Conclusions

This case study and paper solves a highly-constrained real world workforce scheduling problem for engineers at FSO. The solution utilises Greedy Algorithm, Metropolis-Hastings acceptance and variable neighbourhood search to obtain results which would not be obtained using either of these algorithms individually. The greedy algorithm creates an initial solution which is used as a set of neighbourhoods by the VNS to find a closer to optimal solution. The VNS swaps shifts between employees on a single day, or by moving an employee's scheduled shift time from one day to another in the same week, in order to obtain a closer to optimal solution. The greedy algorithm solves the hard constraints, and some soft constraints, but the VNS increases the search space in order to find solutions which improve the number of soft constraints met and improve fitness test results.

Future work will extend the Greedy Algorithm with extra functionality which will provide a starting set of shift patterns which already meet some hard constraints. This head-start should improve the time it takes for the algorithm to run, as well as potentially improving fitness results since closer to optimal patterns can be preset with this method. There is also the option of furthering analysis on which days least meet fitness criteria, and attempting to focus on those days in particular – this may improve fitness more quickly by solving problem areas. The steps to be taken next however will be to create an optimizer which would treat this algorithm as a 'black box', and modifying the input parameters and data in order to look at different and potential scenarios, for example the ability to view the fitness values of solutions created with an increased number of employees, or an increased number of employee skills,

allowing some insight into potential business growth for example. Information such as this could be useful to both employees and businesses who wish to further optimize their skilled workforce.

ACKNOWLEDGMENT

The authors of this paper extend gratitude to those involved and the following funders: British Telecommunications Plc and the UK's EPSRC DAASE project (grant no. EP/J017515/1). We would also like to give our thanks to Paul McMenemy for his assistance with formulating the constraints and his hours of time explaining IP models. Finally we would like to acknowledge Brian Lee for his explanation on analysing non-skewed data.

REFERENCES

[1] E.K. Burke, and K. Graham, Search Methodologies. Springer Science+ Business Media, Incorporated, 2005.

[2] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck, "Personnel scheduling: a literature review," European Journal of Operational Research, vol. 226, pp. 367-385, 2013.

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

[4] J. Li, and U. Aickelin, "BOA for nurse scheduling," in Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications, Chapter 17, M. Pelican, K. Sastry and E. Cantú-Paz, Eds. Springer, 2006, pp. 315-332.

[5] J. Li, E.K. Burke, T. Curtois, S. Petrovic, and R. Qu, "The falling tide algorithm: a new multi-objective approach for complex workforce scheduling," OMEGA – The International Journal of Management Science, vol. 40, pp. 283-293, 2012.

[6] D. Konur,, and M.G. Mihalis, "Cost-stable truck scheduling at a cross-dock facility with unknown truck arrivals: A meta-heuristic approach," Transportation Research Part E: Logistics and Transportation Review, vol. 49, pp. 71-91, 2013.

[7] E.K. Burke, and P. Sanja, "Recent research directions in automated timetabling," European Journal of Operational Research, vol. 140, pp. 266-280, 2002.

[8] J. Li, R. Bai, Y. Shen, and R. Qu, "Search with evolutionary ruin and stochastic rebuild: a theoretic framework and a case study on exam timetabling," European Journal of Operational Research, vol. 242, pp. 798-806, 2015.

[9] M.A. Awadallah, A.L. Bolaji, and M.A. Al-Betar, "A hybrid artificial bee colony for a nurse rostering problem," Applied Soft Computing, vol. 35, pp. 726-739, 2015.

[10] M.J. Brusco, and L.W. Jacobs, "A simulated annealing approach to the cyclic staff-scheduling problem," Naval Research Logistics, vol. 40, pp. 69-84, 1993.

[11] P. Hansen, and N. Mladenović, "Variable neighborhood search," Search methodologies. Springer US, pp. 313-337, 2014.

[12] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[13] P David, and S Ropke, "Large neighborhood search," Handbook of Metaheuristics. Springer US, pp. 399-419, 2010.

[14] E.K. Burke, G. Kendall, Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. Springer, 2014.

[15] E.K. Burke, T. Curtis, G. Post, R. Qu, qnd B. Veltman, "A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem," European Journal of Operational Research, vol. 188, pp. 330-341, 2008.

[16] E.K. Burke, J. Li, and R. Qu, "A Hybrid model of integer programming and variable neighbourhood search for highly-constrained rostering problems," European Journal of Operational Research, vol. 203, pp. 484-493, 2010.

[17] C. Siddhartha, and E Greenberg, "Understanding the metropolis-hastings algorithm," The American Statistician, vol. 49, pp. 327-335, 1995.