Shift Scheduling and Employee Rostering: An Evolutionary Ruin & Recreate Solution

Kenneth N. Reid[1], Jingpeng Li[2], Nadarajen Veerapen[2], Jerry Swan[3], Alistair McCormick[4], Mathias Kern[4], and Gilbert Owusu[4]

[1] University of Stirling, Stirling, UK.
Ken@kenreid.co.uk
[2] University of Stirling, Stirling, UK
[3] University of York, York, UK
[4] BT Research & Innovation, United Kingdom

Abstract. For decades, since the inception of the field, scheduling problems have been solved with a variety of techniques. Many proven algorithms to these problems exist; however, there is no single method to solve all the vast variety of problems that exist across many sub-fields with differing datasets. In this paper we explore the use of an Evolutionary Ruin & Stochastic Recreate algorithm, with a Simulated Annealing control mechanism, to a real-world employee scheduling problem and its ability to solve this problem to near optimality. The combinatorial possibilities of parameterisation are very large - the Taguchi design of experiments method is used to examine a subset of those possibilities within a limited runtime budget. Evolutionary Ruin and Stochastic Recreate has not previously been applied to the specific scheduling domain of employee scheduling and rostering: we investigate the effectiveness of the algorithm with different parameter values and discuss the insight it provides into the runtime effect of the mechanisms of Evolutionary Ruin & Stochastic Recreate.

Keywords: Evolutionary Ruin and Stochastic Recreate, Metaheuristics, Employee Rostering, Shift Scheduling, Simulated Annealing, Design of Experiments,Taguchi Method

1 Introduction

1.1 Summary

In the field of operational research, the optimised allocation of resources to activities is paramount to operating efficiency. Resources in this context can be machines, vehicles, raw materials, employees and so forth. When employees are not working shifts that meet their personal preferences and skill sets there is an associated loss of efficiency. There may be no shifts available that suit the employee (be it due to skill requirements, or shift start times etc) or there may be no employee which suits this shift (similarly, no employee with correct skills, or no employee with a contract that allows allocation to this shift etc.). Traditionally

these have been viewed as two distinct problems and solved independently, in order to reduce the size of the search space. In this implementation the two problems of shift creation and employee allocation are solved simultaneously. While this increases the complexity of the problem, this additionally large search space may assist in finding higher quality results.

In this paper we use the term "shift" to mean a time interval within in some 24 hour period that can have a single employee allocated to it. Shifts have the attributes of start time, end time, length, demand covered (in terms of an integer array with potential values of 0 or 1 per hour), skill covered, and a related shift rule. A shift rule contains requirements set by the company, including minimum / maximum shift lengths, number of shifts, start times, end times. This is described more thoroughly in 2.

The method of Evolutionary Ruin and Stochastic Recreate (ER&SR) is based on the 'Ruin and Recreate' method of Schrimpf et al [15], in which candidate solutions to optimisation problems are partly dismantled by removing certain components (depending on the type of optimisation problem this could be employee shifts, machine schedules, vehicles from rosters, exams, etc). Then the roster (or other type of solution) is reconstructed afterward being ruined. In this paper we show that ER&SR with a Simulated Annealing (SA) control mechanism is able to find close to optimal solutions to shift scheduling and employee rostering problems. Further we measure the internal mechanisms of the algorithm by testing the solution with a variety of hyper-parameterised test cases, designed via the Taguchi method.

To solve this problem, we apply ER&SR first defined in [11] then explored as a theoretical framework in [10]. ER&SR is an algorithm derived from the traditional Ruin and Recreate algorithm defined in [16]. This methodology was chosen as it has thus far been used to solve a complex exam timetabling problem in [10]. More research is necessary in its application in similar problem areas, employee scheduling & rostering being one such area. We chose to use a metaheuristic technique due to the especially large search space being implausible to search for a solution of acceptable quality in a normal business time frame. A metaheuristic can find a near-optimal solution in a reasonable time-frame and has a smaller requirement for domain specific knowledge than exact methods.

The Taguchi method is used as a "design of experiments" approach [14], allowing the parameter space to be explored without need for exhaustive search or an imprecise grid search. This approach is lesser known, particularly in the context of shift scheduling and employee rostering with metaheuristics.

The origin of staff scheduling research was traced to Ernst et al [6] in 1954, where the authors analyse and discuss relief times for toll collectors [5]. In the six decades since this paper, there has been a vast amount of research in the field. Similar staff scheduling problems have been solved with other metaheuristics, such as nurse rostering with hybrid artificial bee colony heuristics [1] and SA for a cyclical staff scheduling problem [3], showing the wide variety of problems in staff scheduling. According to Van den Bergh et al. [2] the single most

popular classes of solution technique for personnel scheduling[5], are Simulation techniques, Integer Programming, Mixed Integer Programming and Constructive Heuristics.

Unlike similar problems tackled previously in the literature, this paper deals with a real-world shift scheduling and employee rostering problem provided by a large international Field Service Operations company (FSO). Thus the constraints, specifications and parameters of the problem are dictated by real world requirements. The benefit of the specification being provided by the FSO is the employee and forecast demand data is a genuine representation of a real problem. Consequently, an empirically grounded view of the algorithm's effectiveness is obtained.

The rest of this paper is structured as follows: in Section 2 the problem description is defined as are the prospective benefits of this approach. Section 3 details the solution implementation in Java, using the theoretical framework provided by [10] as a basis, and the unique elements of this implementation are highlighted. The results of using this approach are presented and discussed in Section 4. The paper concludes with Section 5 where we summarize our research and discuss possible future research directions and provide recognition to funders and partners on this project.

2 Problem Description

In this section, we first describe the problem in detail, followed by the Hard Constraints (HCs) and Soft Constraints (SCs), which describe the requirements of a solution and optional but desired characteristics, respectively.

Employee scheduling is well known to be an NP-Hard problem [7], and our interest is therefore in finding solutions that are 'good enough' to meet FSO time requirements.

Both the shift scheduling and employee rostering problems are typical of organizations which can range from a small number of employees such as nurse rostering [17] to larger scale such as field service engineers [13]. This problem deals with scheduling and rostering a large number of engineers (around 25,000), ranging up to around 200 employees. The problem has the following key characteristics:

1. Due to the real-world data being provided by a UK based FSO, UK Employment Law must be adhered to. This includes maximum working hours allowed per week, number of rest hours in between shifts, etc.
2. The employees described in the data have a variety of skills. Demand is provided for the problem in skill-based manner. Demand may fluctuate greatly day to day, week to week. The objective of meeting demand includes not simply providing manpower to cover number of employees required per shift, but skills required per shift, and per day, while still meeting HCs.

[5] i.e. excluding the most popular but miscellaneous "Other" section

3. The shift creation process is structured by shift rules. Shift rules contain parameters and requirements such as shift length, shift start time boundaries and number of shifts required for this rule.

The problem has the following HCs which must be met for a solution to be deemed feasible:

- HC1: Employees 0 or 1 shifts per day.
- HC2: Employees have a minimum and maximum number of allocations per day of week (Monday - Sunday).
- HC3: Employees can have shift patterns A, B or A/B. A/B means in one week they must do A shifts, in the next B shifts. Shift pattern is synonymous with the number of shifts an employee works in a 7 day week, Monday to Sunday.
- HC4: Employees must be allocated to shifts equal to or after their shift time minimum, and before or equal to their shift time maximum.
- HC5: Employees must be allocated to shift lengths longer than or equal to their minimum shift length allowed, and shorter than or equal to their maximum shift length allowed.
- HC6: Shifts must begin within their shift rules designated start and end times.
- HC7: Shift rules specify a minimum requirement of number of shifts across the scheduling period. A possible minimum is no restriction (-1) or an enforced lower threshold (> 0). Every shift rule also specifies a maximum requirement of shifts across the scheduling period. A possible maximum is no restriction (-1), no shift rules of this type (0) or an enforced upper threshold (> 0).
- HC8: Shifts must be greater than or equal to its shift rules minimum shift length and be lesser than or equal to its shift rules maximum shift length, if specified.

While the above HCs are considered a requirement, the below SCs are preferred but not a requirement of solution feasibility.

- SC1: Shifts should cover the maximum amount of demand requirement they possibly can.
- SC2: Employees should be allocated to shifts that meet their first skill preference.
- SC3: Employees should be allocated to shifts that they meet the skill requirement for.
- SC4: Two consecutive rest days for every employee on every week.

3 Implementation

3.1 Evolutionary Ruin & Stochastic Recreate

In this subsection the ER&SR algorithm is described at a high level, then covered in more detail in subsection 3.2. This research is derived from the implementation described and defined in the theoretical framework [10]. ER&SR itself is

a more advanced variant of the 'Ruin and Recreate' method from which it is derived [15]. ER&SR is executed in four major phases: solution decomposition, evolutionary ruin, stochastic recreate and solution acceptance. This is different from the traditional R&R method with the addition of the new operators Selection and Mutation which are used as the ruin strategies. This provides an evolutionary advantage to this technique, as described in [10].

The Solution Decomposition phase gathers fitness values of the solution as it currently stands per component. A component is a single employee shift allocation object, which contains information on the employee allocated and the shift the employee is allocated to. The advantage of using this so-called 'holistic' approach (as opposed to the more traditional 'perturbative' one) is that by using expert knowledge of the solution space we can provide additional insight into the weaker and stronger components. Some of the constraints however must be considered holistically to resist epistasis. This phase itself only gathers information and does not modify the solution. Each component has a fitness value assigned based on the constraint satisfaction, where failed HCs have a dramatic effect on solution fitness (making it infeasible), whereas SC satisfaction differentiates substandard solutions from better solutions.

The second phase, evolutionary ruin, comprises of two operators: selection and mutation. This phase modifies the state by corrupting the solution, removing components selected in the solution decomposition phase. The selection operator is concerned with the selection based on the fitness of each component. As this is not a deterministic solution — the higher the fitness value the less likely the component is selected for evolutionary ruin. There is still a possibility that even with a high fitness a component can be selected, allowing exploration of the solution space through a controlled stochastic mechanism. The mutation operator is also tasked with removing components from the solution. As implied by the name of the phase, this leaves the solution in a state of intentional deficiency. The solution is deemed to be in a complete state regardless of constraint satisfiability. The wholeness is defined by a complete set of components. The number of components which are mutated is dependent on probabilities derived from the number of assigned and unassigned components from the selection phase. The mutation phase is a useful mechanism for continuing exploration even after a good solution has been derived, so the search process keeps exploring even if the fitness values of the individual components is high, maintaining potential for finding a further improved solution.

Stochastic recreate is the third phase, responsible for generating a new solution state. This phase probabilistically searches through potential succeeding solution states. Stochastic recreate effectively provides stochasticity to the algorithm while still providing favour to the establishment of a fitter solution. There is potential computational difficulty in this phase due to the multifarious nature of nearby solutions. The potential permutations are vast in number, and with a diverse space of components requiring regeneration. This possible problem of exponentiality of neighbours is countered with a rule which states that neighbouring solutions have the same probability of being moved to. This phase is

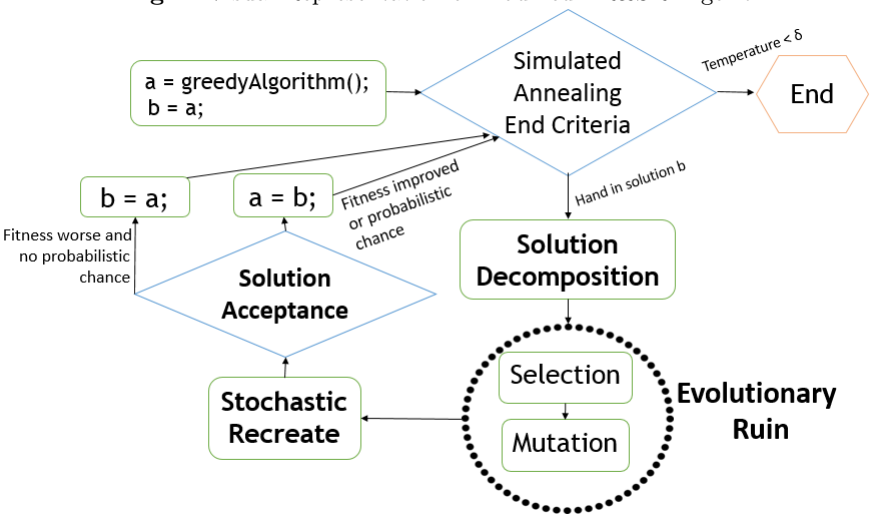
executed repeatedly until the schedule is whole.

The final phase, solution acceptance, accepts one of the previously discovered states based on a conditional probability function between the new state(s) and original. The actual mechanism which controls the solution acceptance phase is problem dependent. The authors of [10] suggest hill-climbing, SA, threshold acceptance, great deluge or falling tide as potential strategies for controlling probability acceptance, or any other strategies which can be found in the literature.

3.2 Application of ER&SR to Shift Scheduling and Employee Rostering Problem

The application of ER&SR to the Shift Scheduling and Employee Rostering Problem followed the approach outlined in section 3.1; however, this section describes the more technical details within the implementation and the changes to the original algorithm required to meet the specific requirements of the problem. The order the algorithm is defined in as specified by the theoretical framework [10] has been modified to meet the requirements of this problem. Further, as recommended in [10], SA is used as the control mechanism to allow a halting operation, and to prevent overt disruption towards the latter stages of runtime. A visual representation of the modified algorithm is given in Fig. 1.

Fig. 1. Visual Representation of Modified ER&SR Algorithm



Listing 1.1. Phase 0 Post-Initialization Greedy Algorithm

```
void GreedyAlgorithm() {
  // Randomize days.
  Shuffle(days)
  for(Day day : days){
    while(areMoreShiftsRequired()) {
     // Initialize with current allocations
      SolutionDecomposition p1 = getFitnessOfAllDayESAs(day)
     // Allocate best employee / shift combination.
     allocateEmployee(fitnessPerAllocationOnDay)
    }
  }
}
```

3.3 Phase 0 - Initialization

The first phase initializes the problem, including creating every potential shift. This means after this phase has run, there is a shift of every potential length, with every potential start and end time, for every particular skill requirement, per shift rule. It is important to note that during initialization, shifts which are impossible are not created (for example no employees work Sundays, so no Sunday shifts are created). This is important as it allows the algorithm to explore the search space without having to recreate shift objects repeatedly, saving on potential processing costs. After initializing, this phase also creates an initial solution by Greedy Algorithm. The purpose of an initial solution is to provide a starting point, reducing potential processing time building a basic solution in the first time. Additionally, it means that the second phase of evolutionary ruin has allocations to ruin during the first cycle. After the initial solution creation, the rest of the algorithm is cyclical until the end criterion is met, but this phase (which can be considered Phase 0) does not repeat. The greedy algorithm in this section is simplistic, and calls on the fitness functions of the Solution Decomposition phase to calculate preferred employee allocations. See listing 1.1 for pseudocode.

3.4 Phase 1 - Solution Decomposition

The solution decomposition consists of Employee Shift Allocation (eSA) objects as the componential elements for analysis. These objects consist of a shift and an employee. Each of the eSA objects across the scheduling and rostering period are assessed for their fitness, which is derived from each objects ability to satisfy the HCs and SCs. The calculation by which the overall object fitness is derived is dependent on the parameterised weightings of each constraint. The SC weightings are parameterised so that users of the algorithm can modify constraints to be worth more or less than other constraints, for use in real-world scenarios (for trying different preferences, techniques etc.). This feature was implemented

for a more realistic and flexible business use case. By default the HC fitness is worth 95% of the total and the final 5% is between the weighted SCs. Due to the nature of this problem there is a separation of holistic constraints and lower level constraints, which are described in the ER&SR algorithm. This is due to the holistic constraints requiring knowledge of the whole solution space, including all shifts and employee allocations. The holistic constraints mentioned are HC2, HC3, HC7 & SC4. In this implementation we deviate from the original algorithm to allow consideration of these holistic constraints. This is achieved by measuring these constraints at a schedule-wide perspective, then modifying the fitness values of individual eSA objects. If an object is found to break these constraints (by, for example, scheduling an employee to work a 6-day week when contracted for 5), then we probabilistically modify the fitness of all offending eSA objects. The schedule thus far is unchanged from the initial solution construction, or since the previous iteration as this solution decomposition phase only provides analysis of solution fitness.

3.5 Phase 2 - Evolutionary Ruin

The next phase is the evolutionary ruin section where every eSA object is evaluated and employees are probabilistically removed from shifts. With all the fitness values obtained from the Solution Decomposition phase, each eSA object is evaluated first in the selection mini-phase. The selection mini-phase first calculates an acceptance rate:

$P = r * x$

Where P is the probability of acceptance, r is a random number between 0 and 1, and x is a parameter to tune the fastidiousness of the selection mini-phase. Then P is compared to the fitness of the object, if the fitness is higher than the random number, continue on to the next eSA. If instead the fitness is lower than P, then remove the employee from this eSA.

After the selection mini-phase is the mutation mini-phase, which will probabilistically remove an employee from every eSA object without regard to fitness. This introduces an additional level of stochasticity by probabilistically deallocating employees from eSA objects, allowing additional exploration of the solution space. This phase solely targets the components which previously were selected for retention in the selection phase (or rather, not selected for deallocation). This proceeds without regard for fitness values. The ruin phase causes the occasional removal of an employee from an individual shift. Careful parameterisation of this phase is important as excessive ruination can hinder exploitation. The chance of the ruination occurring is calculated as follows:

$P = r * x_1$

Where P is the probability, r is a random number between 0 and 1, x is the parameter used to increase or decrease probability, and the subscript is the parameter to control the rate of removal.

3.6 Phase 3 - Stochastic Recreate with Solution Acceptance

This now partially complete (or possibly empty if enough components are destroyed) solution is reconstructed during the stochastic recreate phase. This phase executes repeatedly until a full solution is provided. First, the list of employees is placed into a randomized order to prevent giving preference to the first employees in the list repeatedly. An employee is taken from the list and the number of shifts worked per week is calculated. Every week where the employee is not working enough shifts is allowed to rebuild. This is conducted by selecting a random day in the week the employee is not currently working then selecting a random shift that exists on this day, and the employee is placed onto this shift. The fitness of this component is then analysed, and a random floating-point number between 0 and 5 is generated. Due to the HCs consisting of 95% of the fitness, the generated number becomes the allowed SC fitness. If the allocation is above the acceptable limit $(95\% + r)$ then the allocation is accepted. The recreation method repeats for all employees and all weeks until a full schedule has been generated.

The stochastic recreate with solution acceptance phase is controlled by SA, a frequently used methodology derived from statistical thermodynamics. In this circumstance however it is used to control the ruining capabilities of ER&SR as well as this phase. The acceptance probability is as follows [9]:

$P = exp((o - n)/t) > r$

Where P is the probability of accepting a worsened state, o is the original solution energy cost (modelled as inverse fitness), n is the energy cost of the new solution and r is a random number between 0 and 1. This allows extensive exploration in the early stages of runtime but turns to more exploitive measures later in runtime. If the solution is accepted then the previous solution is overwritten with the current solution, and the algorithm enters the next iteration or ends if the temperature $t < \delta$.If the solution is rejected, the current solution is overwritten with the previous solution, and the algorithm enters the next iterations or ends if the temperature $t < \delta$. This phase has one additional functionality which is noteworthy for industrial usage: due to the nature of SA, the best solution ever found throughout all iterations is not necessarily the solution produced at the end of runtime. As such, this implementation also saves the best solution ever found for actual use.

4 Results

The results in this section are derived from tests which ran on an oracle enterprise Linux 6 OS server with a 24 core 2.3GHz Xeon CPU. Parameters were first tuned using the Taguchi method, some of the results of these tests are also presented in this section. The parameters which provided the best overall fitness (as an unweighted average) are then discussed.

The effectiveness of metaheuristics is largely dependent on well-tuned parameters [4]. There are many experimental designs which can be used for tuning the parameters, such as grid search, random search, or more popular recently;

hyper-heuristics. However the Taguchi method has not been used in personnel scheduling, but has much success in the experimental design of engineering methods [18], electrical engineering [12], designing test cases for a open routing vehicle problem using neural networks [19], and others.

The Taguchi design of experiments evolved from orthogonal arrays and Latin squares [8]. This design of tests allow experimentation with a variety of parameters in order to find a good combination of parameters to produce a high fitness value with ER&SR, and help reduce the testing time (which is deemed important to the FSO).

The cooling rate is used as the SA control mechanism which controls the number of iterations. The temperature is set at 10000 for all tests, and the testing included 6 potential cool rates, from 0.09 - 0.59.

An L16 ($4^{\wedge}2$) orthogonal array was designed consisting of 2 factors with 4 possible values and 16 runs - however as having 0 selection and 0 mutation rate is redundant, this was removed (so only 15 runs). This consisted of the following parameters: the Selection rate (SR) is used during the evolutionary ruin phase as is the mutation rate (MR).

This is combined with an L8 ($2^{\wedge}4$) orthogonal array consisting of 4 factors with 2 possible values and 8 runs. These parameters are simply toggles to turn SCs (SC1 - SC4) on or off. Two tables were used due to the limitations in the number of Taguchi designs that exist.

The parameters the Taguchi tests are designed on described in more detail in 3.2. It is likely other problems would require different parameters, as these parameters are tuned for this problem specification and dataset. The top five fitness results from Taguchi testing are shown in table 1.

Table 1. Top Five Taguchi Test Results

	CR	**SC1**	**SC2**	**SC3**	**SC4**	**SR**	**MR**	**Mean Fitness**
X1	0.29	0	1	1	0	1	1	**97.112**
X2	0.19	0	1	1	0	1	0.66	**97.102**
X3	0.19	0	1	1	0	1	0.66	**97.094**
X4	0.09	0	1	1	0	1	0.66	**97.092**
X5	0.09	0	1	1	0	1	0.66	**96.086**

Testing included running each above Taguchi test 10 times on a demand set of 1 month for close to 200 employees. Fig. 2 highlights a sample of the testing conducted (6 configurations out of 600 total configurations), showing that there is a significant difference in the distribution of fitness levels obtained using different parameter configurations. ANOVA testing was also conducted which confirmed this. However there was little difference between the top 5 results as shown in Fig. 3.

The best case scenario can be seen in more detail in listing 1. The parameters for the highest fitness are fairly interesting, as it seems to be generally preferable to have a maximum amount of selection rate and about 66% of mutation rate.

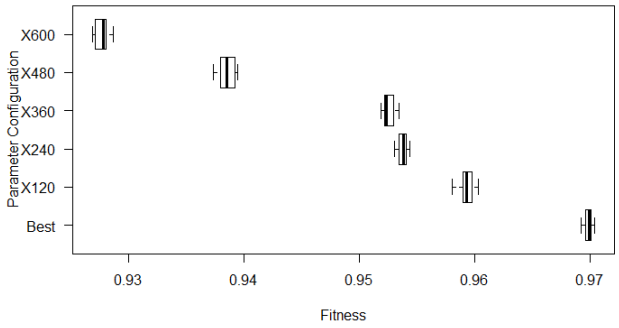


Fig. 2. Box-Plot of Fitness for 6 sample parameter configurations

It also highlights that it is easier to forge a solution where SC2 and SC3 are met than SC1 and SC4. Our recommendation to the FSO in this instance is to test parameterising the soft constraints in a weighted format to find a balance that most satisfies the company's and employees' requirements. An example of the fitness throughout a single test with a single set of parameters can be viewed in Fig. 4.

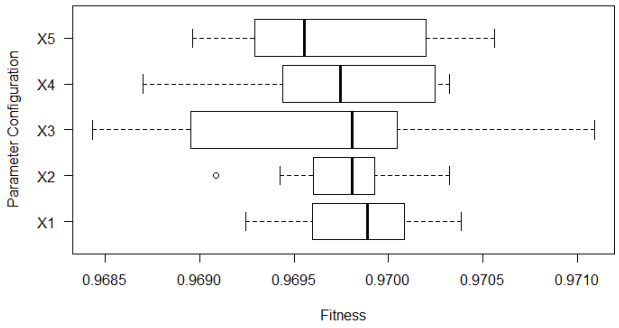


Fig. 3. Box-Plot of Fitness for the best 5 parameter configurations

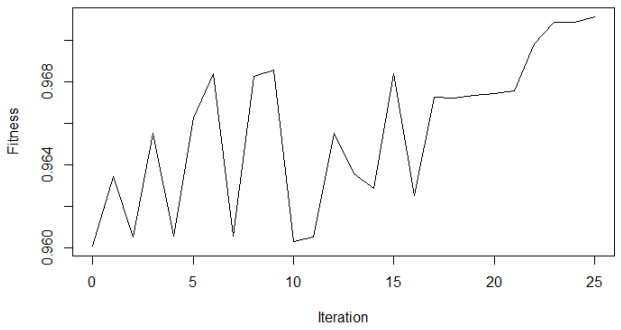


Fig. 4. Fitness per Iteration of a Single Test.

5 Conclusions

In this paper we present a revised implementation of the ER&SR algorithm to tackle a real-world Employee Scheduling and Employee Rostering problem. This approach was selected specifically due to the large search space and inherent complexity of problem employee scheduling problems with large search spaces; exact methods or industry solutions are unlikely to provide near-optimal solutions without much longer running times than this method. However, there is scope for further research using other metaheuristic based approaches to personnel scheduling and employee rostering problems, in particular when compared to industry solvers such as CPLEX.

Acknowledgement. The authors of this paper extend gratitude to those involved and the following funders: British Telecommunications Plc and the UKs EPSRC DAASE project (grant no. EP/J017515/1) J. Swan acknowledges the support of the EU H2020 SAFIRE project (Ref. 723634). We also extend thanks to Dr. Amjad Ullah for his initial explanations of the Taguchi approach and Sarah Thomson for proofreading.

References

1. Awadallah, M.A., Bolaji, A.L., Al-Betar, M.A.: A hybrid artificial bee colony for a nurse rostering problem. Applied Soft Computing 35, 726–739 (2015)
2. Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., De Boeck, L.: Personnel scheduling: A literature review. European Journal of Operational Research 226(3), 367–385 (2013)
3. Brusco, M.J., Jacobs, L.W.: A simulated annealing approach to the cyclic staff-scheduling problem. Naval Research Logistics (NRL) 40(1), 69–84 (1993)

4. Burke, E.K., Kendall, G., et al.: Search methodologies. Springer (2005)
5. Edie, L.C.: Traffic delays at toll booths. Journal of the operations research society of America 2(2), 107–138 (1954)
6. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. European journal of operational research 153(1), 3–27 (2004)
7. Heimerl, C., Kolisch, R.: Scheduling and staffing multiple projects with a multi-skilled workforce. OR spectrum 32(2), 343–368 (2010)
8. Kacker, R.N., Lagergren, E.S., Filliben, J.J.: Taguchi's orthogonal arrays are classical designs of experiments. Journal of research of the National Institute of Standards and Technology 96(5), 577 (1991)
9. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983), http://science.sciencemag.org/content/220/4598/671
10. Li, J., Bai, R., Shen, Y., Qu, R.: Search with evolutionary ruin and stochastic rebuild: A theoretic framework and a case study on exam timetabling. European Journal of Operational Research 242(3), 798–806 (2015)
11. Li, J., Qu, R., Shen, Y.: Evolutionary ruin and stochastic recreate: A case study on the exam timetabling problem. In: ECMS. pp. 347–353 (2012)
12. Mahapatra, S., Patnaik, A.: Optimization of wire electrical discharge machining (wedm) process parameters using taguchi method. The International Journal of Advanced Manufacturing Technology 34(9), 911–925 (2007)
13. Reid, K.N., Li, J., Swan, J., McCormick, A., Owusu, G.: Variable neighbourhood search: A case study for a highly-constrained workforce scheduling problem. In: Computational Intelligence (SSCI), 2016 IEEE Symposium Series on. pp. 1–6. IEEE (2016)
14. Roy, R.: A primer on the taguchi method, competitive manufacturing series. New York pp. 7–80 (1990)
15. Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., Dueck, G.: Record breaking optimization results using the ruin and recreate principle. J. Comput. Phys. 159(2), 139–171 (Apr 2000), http://dx.doi.org/10.1006/jcph.1999.6413
16. Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., Dueck, G.: Record breaking optimization results using the ruin and recreate principle. Journal of Computational Physics 159(2), 139–171 (2000)
17. Shen, Y., Li, J., Peng, K.: An estimation of distribution algorithm for public transport driver scheduling. International Journal of Operational Research 28(2), 245–262 (2017)
18. Taguchi, G., Taguchi, G.: System of experimental design; engineering methods to optimize quality and minimize costs. Tech. rep. (1987)
19. Tyasnurita, R., Özcan, E., John, R.: Learning heuristic selection using a time delay neural network for open vehicle routing. In: Evolutionary Computation (CEC), 2017 IEEE Congress on. pp. 1474–1481. IEEE (2017)