Workforce Rostering via Metaheuristics

Mary Dimitropoulaki(✉), Mathias Kern, Gilbert Owusu, and Alistair McCormick

BT Research and Innovation, Ipswich, UK
{mary.dimitropoulaki,mathias.kern,gilbert.owusu,ali.mccormick}@bt.com

Abstract. Staff scheduling and planning in a cost effective manner has been a topic of scientific discussion for many years, driven by the need of many organisations to fully and effectively utilise their workforce to meet costumer demand and deliver service. Due to the varying nature of industry sectors, problems often require tailoring for particular business needs and types of work. This paper presents an overview of how a version of this problem was solved in a business with a large field workforce. The automation of this process has proved vital in ensuring that there is the right amount of resources rostered in on each day of the week, transforming a lengthy, manual procedure into an operation of a matter of seconds. The paper discusses how a Simulated Annealing approach was implemented, and provides a comparison of its performance versus a standard Hill Climber. We also include a detailed description of how rules and constraints were incorporated into the work, and what effect these had on rostered attendance.

Keywords: Rostering · Heuristic search · Industrial applications of AI

1 Introduction

An abundance of work has been undertaken in the area of automation and improvement of workforce rostering in areas such as call centres, health care systems, civic services & utilities and hospitality & tourism [8]. The diverse and complex dynamics of problems of this type mean that generalisation can prove difficult. Maximising operational effectiveness is known to lead to better customer experience and compliance to service level agreements. Prior to the automation of scheduling, one of the first steps to achieve this is to optimise resourcing and planning processes. Ensuring that there is a sufficient resource available on each day is key to successful field service management, and therefore the automation and optimisation of this process is essential. This work presents a solution to a rostering process problem of a large workforce of field resources via the use of metaheuristic methods.

The process of rostering resources is performed by a dedicated team and consists of the following manual steps, which can require days to complete:

– Create roster patterns

M. Bramer and M. Petridis (Eds.): SGAI-AI 2018, LNAI 11311, pp. 277–290, 2018.
https://doi.org/10.1007/978-3-030-04191-5_25

– Allocate resources to roster patterns
– Update allocation and implement roster assignments

These are building blocks of employee rostering, but the challenges occur with workload prediction, staffing requirements, shift design and allocation [15]. An important factor to note is that each business' operational requirements such as laws, planning periods and shift types can make this process very complex, and thus solutions ought to be tailored accordingly [15]. This paper will focus on assigning the ideal start week within their roster pattern for each resource. Rosters in this context are cyclical and we will be imposing a set of modifiable constraints (rules) with varying objectives. We are optimising the attendance across teams of approximately 100 to 300 field resources in a typical scenario. Roster patterns usually vary between 5 and 25 weeks in length, meaning each field resource could be on one of 5 to 25 positions within their pattern. This means there is a very large number of possible combinations of where the 100 to 300 resources are within their roster patterns. Over time attendance can become unbalanced, leaving some days with more field resources than needed and others with not enough. Shifting resources within their roster pattern will re-balance, but it is important to take as well into account the aim to minimise changes to individual resources and to ensure rules compliance.

The most studied and well-known rostering optimisation is the nurse rostering problem, which has been solved using many approaches and methods. Some of these are based on heuristics such as Tabu Search, Simulated Annealing and Genetic Algorithms [5]. A comparative study of solving the NRP performed by Kundu et al. [11] shows that constraints are best enforced through weightings, something which will be incorporated into our work and discussed in Sect. 1.1. They found that Simulated Annealing out-performed the Genetic Algorithm, with the most efficient roster being cyclic. In Hadwan and Ayob [10], we also see a similar approach in which shifts are created and then optimised via the use of Simulated Annealing with constraints introduced as weightings. A theoretical discussion can be found in Ernst et al. [8] who noted that the demands of different businesses result in different approaches when it comes to the automation of rostering. They discuss that the modules required to construct a roster vary from organisation to organisation, as some areas require demand modelling and others do not require task assignment. In our scenario task assignment cannot be done in advance, and thus after the roster design phase it is vital to assign the roster patterns to resources effectively. Once this stage has been completed, the best combination of start weeks for the field resources within their roster patterns is required for a balanced attendance. Scheduling and rostering in most cases are interrelated and a combined solution can be seen as more effective, with an example of this being achieved in [14] where this is applied to the driver rostering problem. When it comes to very large organisations, it is almost impossible to combine these two problems and thus these should be separated procedures. The sheer size of the workforce available in our scenario does not allow individual shift allocation, and further legal constrains and agreements limit the scope for regular changes of roster pattern. In our problem scenario, we consider the

roster pattern of each resource as given and fixed but the actual position of each resource within their roster pattern can be changed, and our aim is to find the best current assignment of these positions to achieve the best overall attendance balance over a certain period into the future.

Airline crew rostering is another widely studied scheduling problem. This is an area in which we see most benefits in the automation of scheduling, with almost a $50 million dollar cost saving per year for large airlines [1]. The most common methods rosters are assigned to crew is via bidline systems, personalised rostering or bidding systems. However many challenges remain unsolved such as crew pairing and fleet assignment [1]. Thiel [12] presents an interesting solution to crew pairing through team oriented rostering, a method which enhances team stability and team work quality, but the large amount of rosters and combinations were found to be major issues. In our case the same problems arise as human decision makers usually cannot deal with the amount of data available, and therefore a tool supporting such procedures is vital in more complex organisations [12]. Other public transport rostering has been studied in [4,7,9] in which we see applications in railway crew management, and in [14,16] where the driver rostering problem in public bus transportations is investigated.

1.1 Problem Overview

In this our problem scenario, we consider individual geographical areas which contain between 100 to 300 resources. Each of these geographical areas is divided into 4 to 10 sub-areas. The typical number of roster patterns used by the resources in each area can vary, but is typically below 15.

Roster patterns run over a given sequence of weeks. Upon the assignment of resources to a roster pattern, they will be given a start week number specifying their starting point within the roster pattern cycle. For example, a 13-week pattern means that if a resource starts working on week 9 of this pattern, they will continue working on week 10, 11, 12 and 13 followed by weeks 1 to 8. Afterwards they will repeat this cycle with weeks, 9, 10 and so on.

Roster patterns can also contain certain types of sub-cycles which determine the number of days worked in each week. For example, a resource could be on a 14 week roster pattern with 7 2-week sub-cycles, with the resource working 4 days in the first week and 5 days in the second week of each sub-cycle, i.e. working 9 days across each 2-week sub-cycle.

Over time, attendance across a longer time period can become unbalanced. For example over a 13-week period, one Saturday can see 40 working engineers while another Saturday can only have 25 rostered engineers. These different Saturday starting positions would pose a serious challenge for the operational teams, and a more stable figure of approximately 33 engineers each Saturday would be beneficial. Such imbalances are driven by resources leaving and joining, and by moving to different roster patterns. Proportionally, such imbalances can be even higher when looking at particular sub-areas or certain skilled cohorts of resources.

We have developed an approach, and tool, to aid the process of rostering in resources in order to balance attendance (a) at the overall area level and (b) also within certain sub-area and skill. The ultimate aim is to keep the variation of resource attendance seen across all Mondays over a certain future time period to a minimum, and Tuesdays, and Wednesdays, and so on.

As mentioned previously, we impose a set of constrains when searching for the optimal combination of roster start weeks:

(A) Hard Constraints
- Optimise attendance at skill & sub-area level by grouping resources by this criteria.
- Resources can only move to weeks that satisfy the completion of the sub-cycles. If a resource is currently on a 4-day week in their sub-cycle, we can only move them to another 4-day week of a different sub-cycle.
- A specified gap between working Saturdays must be ensured.
- Restrict percentage of resources allowed to change start weeks in order to minimise disruption and impact on their schedules.

(B) Soft Constraints[1]
- We group engineers by skill & sub-area to ensure optimisation both for the overall area and across these lower level cohorts, and use weights to determine which groups to focus on.
- Days like Saturdays are often more volatile due to the smaller number of resources working. We thus prefer to see Saturdays more optimised, which is ensured by a higher weighting of the Saturday resource balance compared to other weekdays.

(C) Cost function[2]

The measure of attendance is given by the difference between the minimum and maximum number of employees attending on a single weekday over the span of all weeks in the roster pattern (e.g. all Mondays). From now on this will be referred to as the **range**, and so our main goal is to minimise the range for every day of the week.

Assume we have an attendance matrix, $\boldsymbol{A}$, of dimension $n \times 7$, where the rows, i, represent the week numbers and the columns, j, represent the days. This matrix is represented as

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{17} \\ a_{21} & a_{22} & \cdots & a_{27} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n7} \end{pmatrix}. \tag{1}$$

We take the $\min \boldsymbol{A}$ and $\max \boldsymbol{A}$ for each column j, $1 \leq j \leq 7$, to calculate the relativity factor between these two values. This would be given by

$$\frac{\text{Range}_j}{\max_j \boldsymbol{A}} = \frac{\max_j \boldsymbol{A} - \min_j \boldsymbol{A}}{\max_j \boldsymbol{A}},$$

[1] Emphasis on specific areas to ensure targeted optimisation through the use of penalties/weights.

[2] The cost function is dependant on the different types of soft constraints.

and so for example on a weekday where we have 60 as the maximum number of resources and 50 as the minimum, the relativity would be 16%. The score for each group of employees is given by

$$cost = \sum_{1 \leq j \leq 7} w_j \left(\frac{\text{Range}_j}{\max_j \boldsymbol{A}} \right)^2, \tag{2}$$

in which the relativity has been squared, resulting in smoother range values when optimising. w_j are weekday-specific weights.
The overall cost function is then given by the sum of all the scores,

$$Overall\ group\ cost = \sum_{i=1}^{G} (cost_{group_i}), \tag{3}$$

where G is the number of resource groups, or cohorts.
As mentioned previously, we have to account for volatile weekdays such as Saturdays where the number of attending resources is significantly less than on other days of the week, resulting in a higher relativity factor. To prevent the score from being affected by the larger relativity, we multiply the cost function by a penalty/weight for each day of the week.[3] For example, we can use an attendance weight vector like $w = (1, 1, 1, 1, 1, 10, 10)$ to focus particularly on minimising Saturday and Sunday ranges.
The next step is to split this amongst the resource groups or cohorts. Recall, we have a single group corresponding to the attendance of all resources across the entire area and groups/cohorts consisting of resources sharing the same sub-area and/or skill level. In order to ensure well-balanced optimisation of ranges overall and within the smaller groups, we weigh the impacts of these two elements differently. We have achieved good results with a 40% weight for the optimisation across the overall areas and a 60% weight for optimising the smaller resource cohorts.

2 Methodology

The type of optimisation problem discussed in this paper is combinatorial as there is a finite set of solutions. Due to the size of the solution space it is not possible to search it exhaustively, and instead we are searching for good solutions close to the maximum fitness in the solution space. We have implement and tested two different techniques, Hill Climbing and Simulated Annealing. Heuristic approaches have been popular when it comes to crew rostering and planning, such as in [2,3,13]. As previously mentioned and commonly discussed in the literature, organisations have different requirements and hence a unique implementation is often necessary for the specific problem scenario.

[3] For example, a range of 10 on Saturday should not be considered the same as a range of 10 on any other day of the week due to the smaller number of attending resources.

2.1 Simulated Annealing

Simulated annealing is a widely used optimisation method derived from statistical thermodynamics. Annealing is the process of heating metals beyond melting point and then by cooling it slowly until solidified into a crystalline structure [6].

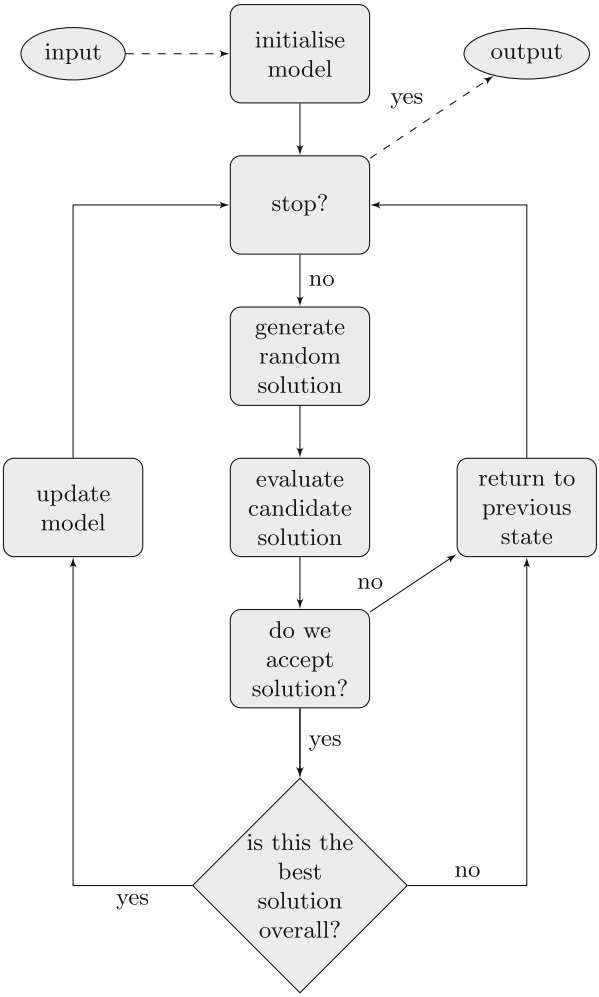

Fig. 1. Simulated annealing algorithm

An overview of the algorithm used in this tool is shown in Fig. 1. We start the process by initialising the model using attendance data, e.g. a 13-week attendance period for all resources starting from a specified date. We build this attendance picture both overall for all resources and for each individual group/cohort

of resources. We evaluate the fitness of the current solution, f_c, and then choose a non-fixed[4] resource at random to set their start week to a random week[5]. Let f_n denote the value of the new fitness. Let s and s^* denote the current and neighbouring state/solution, respectively. The acceptance of states depends on the value of the new fitness given after evaluation, but also the *acceptance probability*, a function obtained through

$$P\big(f_n, f_c, T\big) = e^{\frac{f_c - f_n}{T}}, \tag{4}$$

where T denotes the temperature of the system, a global time-varying parameter used as a stopping condition. Stopping conditions in simulated annealing depend on the *cooling schedule* implemented.

In order to gain a deeper understanding as to how simulated annealing algorithms are formed, we will discuss the physical interpretation of this process. If $f_n(s^*)$ and $f_c(s)$ are the "internal energies" of each state, these are equivalent to the energies of a physical system. Ultimately, we wish to find the minimum $f_n(s^*)$ of the system. We see in Eq. 4 that the acceptance probability P depends on both energies and the temperature T. We require P to always be positive, hence the method does not get stuck in a local minimum that is worse than the global one. When $T \to 0$,

$$P\big(f_n, f_c, T\big) \begin{cases} \to 0, & \text{for } f_n(s^*) > f_c(s) \\ > 0, & \text{otherwise} \end{cases}$$

Hence, for sufficiently small T, the system increasingly favours moves that minimise the energy function and avoids moves that increase this. This is where the cooling schedule becomes important to the formulation of this method. A cooling schedule mainly depends on the cooling rate, which should be low enough for the probability distribution to always be near equilibrium. Simulated annealing is used to solve a wide range of problems and thus cooling schedules tend to be unique and tailored for each requirement.

The cooling schedule in our implementation starts with an initial temperature of $T_0 = 1$. After each iteration we decrease it by multiplying with a cooling rate, $c = 0.99995$ (i.e. $T = T \cdot c$) while $T > 0.01$. This results in approximately 92,101 iterations.

Note that, due to the fitness function described in Eq. 2, the highest fitness/internal energy we can obtain is 0. To account for this fact Eq. 4 is replaced with

$$P\big(f_n, f_c, T\big) = e^{\frac{f_n - f_c}{T}}. \tag{5}$$

This means that we are looking for the maximum energy f_n our system can achieve and so for P to be positive, $f_n(s^*) < f_c(s)$.

For each T we generate a random number $r \in [0, 1]$, and so solutions only get accepted if $P > r$ or $f_n > f_c$.

[4] A "fixed" resource simply means that they cannot change start week.

[5] Note, some resources are only allowed certain start weeks due to constraints related to factors such as sub-cycles and consecutive Saturdays.

Now let f_o be the best score found over all iterations. If $f_n > f_o$, we set $f_o = f_n$ and update the attendance by adding the new start week for the resource chosen at random. If solutions do not get accepted, i.e. $f_n \leq f_o$, we return to previous state and generate another random solution, until temperature reaches the minimum value set.

In Fig. 2 we see how the cooling rates affect the value of the fitness. As expected, for larger cooling rates we see the value of the maximum fitness converging to zero. This is due to the fact that for larger cooling rates, the temperature takes longer to converge to the value of 0.01 (i.e. the minimum temperature of the system chosen and thus the stopping condition) and so the number of iterations increases.

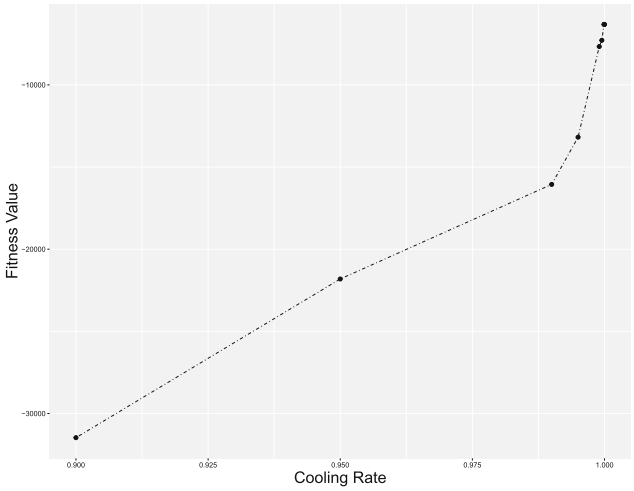

Fig. 2. Maximum fitness values against different cooling rates for $T_0 = 1$ and $T_{min} = 0.01$.

By looking at the best fitness achieved for each iteration we see that f_o starts converging to zero above 20,000 iterations which can be observed in Fig. 3. We also note that between ≈5,000 and ≈20,000 iterations, there is no significantly better solution found and thus the fitness value stops increasing. Note that the example used is on a specific set of test data, but we have observed the same general behaviour for a number of test scenarios.

3 Results

As mentioned previously, to solve this problem we initially implemented a Hill Climbing technique. In this section we discuss and compare the effectiveness of both the basic Hill Climber and the Simulated Annealing approach. We will also be observing how different constraints affect the results and discuss what

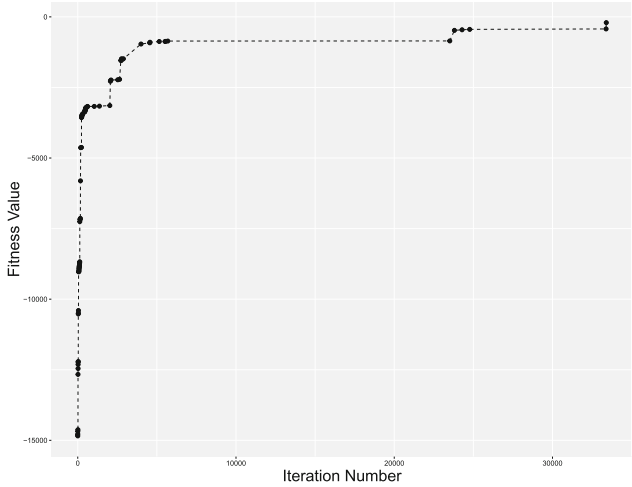

Fig. 3. Best overall fitness, f_o, against iteration numbers with $T_0 = 1$, $T_{min} = 0.01$ and $c = 0.99995$.

compromises, or choices, one can make when addressing a problem of such complexity.

3.1 Hill Climber vs Simulated Annealing

Hill Climbing is a basic local search optimisation technique: we look for the best solution in the neighbourhood of solutions similar to the current one, and do not accept any move that worsens the overall fitness (in contrast to Simulated Annealing). While the first run of the Hill Climber started from the original solution, we allowed for for a number of randomised restarts thereafter.

In general, we see that both methods significantly optimise, i.e. balance, resource attendance. For testing purposes, we will examine how these methods compare for a particular problem scenario, based on including the following constraints:

- Engineers are grouped by sub-area and skill, with appropriate weighting added (i.e. 60% of weight value split within groups and 40% for the overall area).
- Saturdays are weighted with a penalty $\times 10$.
- Potential new start weeks obey sub-cycle rules.

By running the tool for the overall area, we get attendance results that look like the example shown in Table 1. We see the maximum & minimum number of resources attending on each day of the week over a certain time period, as well as their difference, the range. Note that, on Saturday we see a much lower attendance, and so a range of 11 cannot be considered the same as a range of 14

Table 1. Overall attendance before optimisation.

Day	Maximum resources	Minimum resources	Range
Monday	172	158	14
Tuesday	192	178	14
Wednesday	200	194	6
Thursday	181	163	18
Friday	187	171	16
Saturday	37	26	11
Sunday	0	0	0

on Monday. In Table 3, we can view a breakdown of the ranges for some of the sub-areas and skills (in this example we had 7 sub-areas). Note that skills are split into skill level X and skill level Y.

Table 2. Optimised ranges for the overall area produced by Simulated Annealing and the Hill Climber.

Day	Simulated Annealing	Hill Climber
Monday	3	4
Tuesday	5	6
Wednesday	3	4
Thursday	2	5
Friday	2	4
Saturday	1	1
Sunday	0	0

A good way to view the results after running the algorithm is to look at the new ranges across the weekdays. In Table 2, we see the improved ranges for the overall domain, and corresponding views exist for the smaller resource groups determined by sub-area and/or skill. Both the Hill Climber and Simulated Annealing produce good results in our example scenario in the overall domain, but especially so at individual skills and work areas (see Table 3). Simulated Annealing is generally more effective, as our experiments over a larger number of problem scenarios have shown, in particular when more constraints were added. It is well known that Hill Climbers can get more easily stuck in a local optimum, whereas the Simulated Annealing approach allows for escaping such local optima.

In Table 3, we see the optimised ranges in the example sub-areas and skill groups after both techniques, indicating that grouping by skill and sub-area optimises both sub-area and skills. Simulated Annealing with constraints will be discussed in more detail in the next section.

Table 3. Ranges pre-optimisation and after optimisation using Simulated Annealing and Hill Climbing in sub-areas and skills.

Day	Sub-area A			Sub-area B			Skill level X			Skill level Y		
	Pre	SA	HC	Pre	SA	HC	Pre	SA	HC	Pre	SA	HC
Monday	5	3	4	5	3	4	8	3	4	7	3	3
Tuesday	4	4	5	2	3	5	7	4	6	5	3	3
Wednesday	2	1	2	1	2	2	6	2	2	5	3	3
Thursday	5	4	6	3	5	3	10	2	5	8	2	3
Friday	5	5	5	4	3	4	11	3	3	8	2	4
Saturday	4	2	6	4	3	3	8	1	1	6	1	1
Sunday	0	0	0	0	0	0	0	0	0	0	0	0

By comparing the best fitness values of the two algorithms for 20 different geographical areas, i.e. problem instances, as shown in Fig. 4, we see that Simulated Annealing produces better scores in the majority of scenarios. The two cases in which this was not true, the randomised restarts of the HC allowed it to start from more promising initial solutions. Given more time, we expect the SA

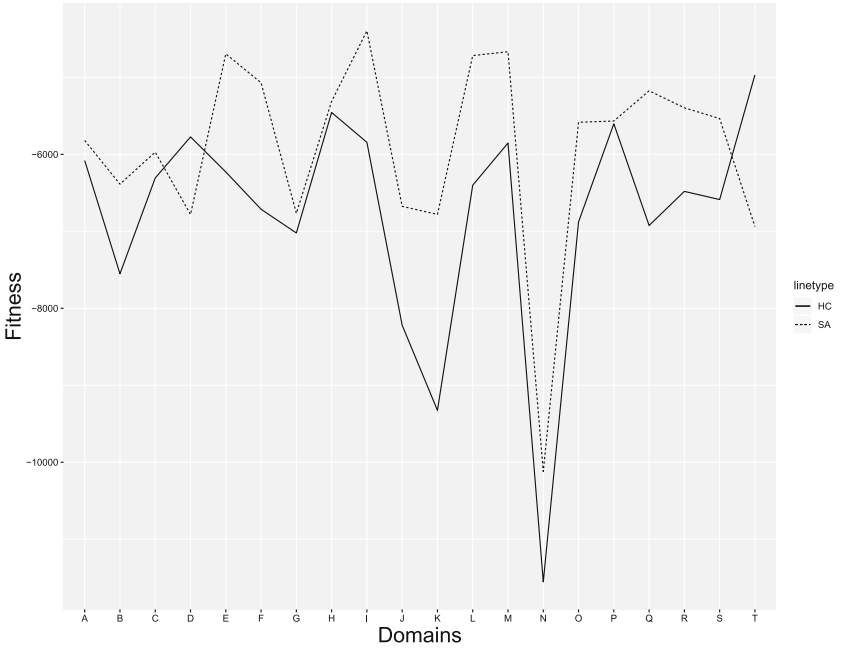

Fig. 4. Best overall fitness values produced for 20 domains by simulated annealing and hill climbing.

algorithm to match or better the HC solutions in these scenarios as well, however our SA solution was balanced for both speed and solution quality. Therefore we will be focusing on the SA approach going forward.

3.2 Optimising with Constraints

In this section we will discuss the results produced by the Simulated Annealing approach when applying the constraints discussed in Sect. 1.1. One of the hard constraints is that a break between working Saturdays must be enforced, ensuring a fair distribution of working Saturdays across all resources. In other words, roster week changes for an individual resource must not result in a resource working on a Saturday too soon after the previous working Saturday. We have also studied how to set the proportion of engineers for which the starting week can be changed, and how low this proportion can be before solution quality suffers significantly. For example, a 40% threshold means that no more than 40% of engineers can change start week.

In our first set of experiments, we have averaged results over 20 scenarios (geographical areas). We vary the aforementioned threshold values while maintaining a Saturday gap of 2 weeks (i.e. no work on two consecutive Saturdays). Table 4 shows that ranges improve - reduce - significantly from a threshold value of 10% to a value of 30%, but further reductions are rather more limited for larger thresholds. For practical purposes, we therefore recommend a threshold value of 30%, i.e. the algorithm will at most change 3 out of 10 resource's roster pattern week.

Table 4. Average of ranges for a number of skills and sub-areas for different threshold values.

Threshold	Domain	Sub-area A	Sub-area B	Skill level Y
10	4.00	3.71	4.57	5.58
30	2.43	2.17	4.57	4.57
50	2.29	2.14	3.42	3.85
70	2.29	2.14	3.7	4.28

In our second set of experiments, we keep the threshold constant at 30% but vary the size of the allowed Saturday gap. Enforcing a larger gap between working Saturdays - 3 weeks - leads to a significantly worse average range at domain level. However there is little difference between a gap of one or two weeks, and thus we recommend a value of two weeks, ensuring no resource is required to work two consecutive Saturdays at the point when the new improved overall solution is implemented (see Table 5).

Table 5. Average of ranges for a number of skills and work areas for different Saturday gap values.

Saturday gap	Domain
1 week	2.42
2 weeks	2.43
3 weeks	3.71

4 Discussion

In this paper, we introduced and described a rostering problem for a large workforce of field resources, discussed two solution techniques - a Hill Climber and a Simulated Annealing approach, and analysed and compared their performance. It is vital to understand and handle the specific business requirements in order to meaningfully address such real-life problem scenarios. Both algorithms consistently improved the initial roster setup in all test scenarios, and Simulated Annealing produced superior results over the more basic Hill Climber in the majority of cases. Furthermore, we showed how the amount of necessary (roster pattern week) changes can be limited without impacting the quality of the overall solution too much, thereby striking a balance between the need of the business to find better roster setups and the desire of individual field resources for minimal disruption.

The output of our approach has been used to improve the roster pattern setup across a large workforce of several thousand field resources. As a result, a more stable basic resource attendance has been achieved, meaning the fluctuations in attendance from one week to the next have been reduced. The greatest impact has been seen on Saturdays where we now see more similar numbers of available resources each week, rather than alternating weeks of too much or too little resource. This, in turn, has positively impacted - simplified - the resource planning process, and the need to cover supply shortages with overtime has diminished.

Going forward, we are planning to focus our research and development efforts on four key areas. Firstly, we want to fully automate the optimisation process so that it runs regularly and automatically flags if the quality of the current roster setup has declined too much. Secondly, we would like to strengthen the fairness aspect of our approach, ensuring that changes to roster pattern start weeks are shared equally among field resources over a longer period. Thirdly, we want to extend our approach from changing the start week within a fixed roster pattern per resource to an approach where we can also consider changing the roster pattern itself for a resource. And finally, we would like to further assess the performance of the chosen Simulating Annealing solution by comparing it to more alternatives such a Genetic Algorithm (GA) and Greedy Randomized Adaptive Search Procedure (GRASP) approaches. All four developments will enhance the applicability of our proposed solution in real-life scenarios.

References

1. Barnhart, C., Cohn, A.M., Johnson, E.L., Klabjan, D., Nemhauser, G.L., Vance, P.H.: Airline crew scheduling. In: Hall, R.W. (ed.) Handbook of Transportation Science. International Series in Operations Research & Management Science, vol. 56, pp. 517–560. Springer, US (2003). https://doi.org/10.1007/0-306-48058-1_14
2. Brusco, M.J., Jacobs, L.W.: A simulated annealing approach to the cyclic staff-scheduling problem. Nav. Res. Logist. (NRL) **40**(1), 69–84 (1993)
3. Burns, A., Hayes, N., Richardson, M.F.: Generating feasible cyclic schedules. Control Eng. Pract. **3**(2), 151–162 (1995)
4. Caprara, A., Fischetti, M., Toth, P., Vigo, D., Guida, P.L.: Algorithms for railway crew management. Math. Program. **79**, 125–141 (1997)
5. Cheang, B., Li, H., Lim, A., Rodrigues, B.: Nurse rostering problems - a bibliographic survey. Eur. J. Oper. Res. **151**(3), 447–460 (2003)
6. Du, K.L., Swamy, M.N.S.: Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature. Birkhäuser, Basel (2016)
7. Ernst, A., Krishnamoorthy, M., Dowling, D.: Train crew rostering using simulated annealing. In: Caccetta, L., Teo, K.L., Sieq, P.F., Leung, Y.H., Jennings, L.S., Rehbock, V. (eds.) Proceedings of International Conference on Optimisation Techniques and Applications, pp. 859–866 (1998)
8. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: a review of applications, methods and models. Eur. J. Oper. Res. **153**(1), 3–27 (2004)
9. Gonçalves, R., Gomide, F., Lagrimante, R.: Methodology and algorithms for railway crew management. IFAC Proc. Vol. **33**(9), 323–328 (2000)
10. Hadwan, M., Ayob, M.: A constructive shift patterns approach with simulated annealing for nurse rostering problem. Proceedings of 2010 International Symposium on Information Technology - Visual Informatics, ITSim 2010, p. 1 (2010)
11. Kundu, S., Mahato, M., Mahanty, B., Acharyya, S.: Comparative performance of simulated annealing and genetic algorithm in solving nurse scheduling problem. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, p. 1 (2008)
12. Thiel, M.P.: Team-oriented airline crew scheduling and rostering: problem description, solution approaches, and decision support. Ph.D. thesis, Faculty of Business Administration and Economics at the University of Paderborn, Germany (2005)
13. Thompson, G.M.: A simulated annealing heuristic for shift scheduling using non-continuously available employees. Comput. Oper. Res. **23**(3), 275–288 (1996)
14. Valdes, V.A.V.: Integrating crew scheduling and rostering problems. Ph.D. thesis, Alma Mater Studiorum Universita di Bologna, Italy (2010)
15. Voudouris, C., Owusu, G., Dorne, R., Lesaint, D.: Service Chain Management: Technology Innovation for the Service Business. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-75504-3
16. Xie, L., Kliewer, N., Suhl, L.: Integrated driver rostering problem in public bus transit. Procedia Soc. Behav. Sci. **54**, 656–665 (2012)