Preprint from *Research and Development in Intelligent Systems* XXXI (2014) pp 211-223

# Rail-Freight Crew Scheduling with a Genetic Algorithm

**E. Khmeleva[1], A. A. Hopgood[1], L. Tipi[1] and M. Shahidan[1]**

**Abstract** This article presents a novel genetic algorithm designed for the solution of the Crew Scheduling Problem (CSP) in the rail-freight industry. CSP is the task of assigning drivers to a sequence of train trips while ensuring that no driver's schedule exceeds the permitted working hours, that each driver starts and finishes their day's work at the same location, and that no train routes are left without a driver. Real-life CSPs are extremely complex due to the large number of trips, opportunities to use other means of transportation, and numerous government regulations and trade union agreements. CSP is usually modelled as a set-covering problem and solved with linear programming methods. However, the sheer volume of data makes the application of conventional techniques computationally expensive, while existing genetic algorithms often struggle to handle the large number of constraints. A genetic algorithm is presented that overcomes these challenges by using an indirect chromosome representation and decoding procedure. Experiments using real schedules on the UK national rail network show that the algorithm provides an effective solution within a faster timeframe than alternative approaches.

## 1 Introduction

While international trade continues to expand, businesses are striving to increase reliability and reduce their environmental impact. As a result, demand for rail freight increases every year and rail-freight carriers attempt to maximize their efficiency. The crew cost constitutes 20-25% of the total rail-freight operating cost and is second only to cost of fuel. Therefore even a small improvement in the scheduling processes can save a company millions of pounds a year.

The CSP in the rail-freight industry is the problem of constructing a schedule for a train driver. Each schedule contains instructions for the driver of what he or she should do on a particular day. Within the industry, the driver's schedule is called a *diagram*. Each diagram should cover all the trains driven by a driver in a

[1] Sheffield Business School, Sheffield Hallam University, Howard Street, Sheffield S1 1WB, UK.

e.khmeleva@shu.ac.uk; a.hopgood@shu.ac.uk; l.tipi@shu.ac.uk; m.shahidan@shu.ac.uk

given day. It must start and end at the same station and obey all labour laws and trade union agreements. These rules regulate the maximum diagram duration, maximum continuous and aggregate driving time in a diagram, and minimum break time.

All drivers are located in *depots* where they start and finish their work. Depots are distributed fairly evenly across the UK. Sometimes in order to connect two trips that finish and start at different locations, a driver has to travel on a passenger train, taxi or a freight train driven by another driver. The situation of a driver travelling as a passenger while on duty is called *deadheading*. The cost of deadheading varies and depends on the means of transportation and business agreements between operating companies. Despite the potential cost, deadheading is sometimes inevitable and it can benefit the overall schedule [1].

Due to employment contract terms, the drivers are paid the same hourly rate for any time spent on duty regardless of the number of hours they have actually been driving the train. Moreover, in accordance with collectively bargained contracts, each driver has a fixed number of working hours per year, so the company is obliged to pay for all the stated hours in full even if some of the hours are not utilized. Paid additional overtime hours can be worked at the driver's discretion. Thus it is in the best interests of the company to use the agreed driving hours in the most efficient and economical way.

Taking all of this into consideration, the operational objectives for the diagrams are:

1. Minimize a number of unused and excess contract hours at the end of the year with a minimum spread of durations of the diagrams. All diagrams will therefore be of duration close to the average 8.5 hours, i.e. the annual contract hours divided by the number of the working days.

$$T_{diagram} = T_{driving} + T_{deadheading} + T_{break} + T_{idle}$$
$$T_{diagram} \rightarrow T_{average}$$

2. Maximize the throttle time, i.e. the proportion of the work shift that is actually spent driving a train. It excludes time for deadheading and waiting between trips.

$$Throttle\ Time = \frac{T_{driving}}{T_{diagram}}$$

## 2 Approaches to crew scheduling

The CSP is usually solved in two stages. At the first stage, all possible diagrams satisfying the industrial constraints are enumerated. At the second stage, only the set of diagrams that covers the entire schedule in the most cost-effective way is identified. Diagrams are usually modelled as binary vectors (Figure 1) where '1' denotes that the trip *i* is included in the diagram *j*, otherwise '0' is inserted. Each diagram has its own cost. The deadhead journeys are displayed by including the same trip in more than one diagram. In the rest of the article the terms diagram and column will be used interchangeably.

| | Diagram1 | Diagram2 | Diagram3 | Diagram4 |
|---|---|---|---|---|
| Trip1 | 1 | 0 | 0 | 1 |
| Trip2 | 0 | 1 | 1 | 0 |
| Trip3 | 0 | 1 | 0 | 1 |
| Trip4 | 0 | 1 | 0 | 1 |
| Trip5 | 1 | 1 | 0 | 0 |

Figure 1 Diagrams

Although the generation of the diagrams can be performed in a simple and relatively straightforward manner using various graph search and label-setting techniques [2], finding an optimal set of diagrams may be highly time-consuming. The problem boils down to the solution of the 0–1 integer combinatorial optimization set covering problem (SCP):

$$Minimize \sum_{j=1}^{m} c_j x_j$$

$$Subject\ to: \sum_{i=1}^{n} a_{ij} x_j \geq 1$$

$$x_j \in \{0,1\}$$

$$i = 1,2 \ldots n\ trips$$

$$j = 1,2 \ldots m\ diagrams$$

where $a_{ij}$ is a decision variable indicating whether a trip *i* is included in the diagram *j*; $x_j$ shows if the diagram is included in the schedule; *cj* is the cost of the diagram.

## *2.1 Branch-and-price*

The complete enumeration of all possible diagrams is likely to be impractical due to the large geographical scope of operations, the number of train services, and industry regulations. Linear programming methods such as branch-and-price [3, 4] have been popular for the solution of medium-sized CSPs in the passenger train and airline industries [5]. These methods usually rely on a column-generation approach, where the main principle is to generate diagrams in the course of the algorithm, rather than having them all constructed a priori. Despite the ability of the algorithm to work with an incomplete set of columns, the column generation method alone does not guarantee an integer solution of SCP. It is usually used in conjunction with various branching techniques that are able to find the nearest integer optimal solution. However, this approach is not quite suitable for the CSP in rail freight, where the possible number of diagrams tend to be considerably higher.

## *2.2 Genetic algorithms*

Linear programming (LP) has been used for CSP since the 1960s [6 ] but genetic algorithms (GAs) were introduced more recently [7]. GAs have been applied either for the production of additional columns as a part of column generation [6] or for the solution of SCP from the set of columns generated prior to the application of a GA [9-12], but there are not yet any reports of them solving both stages of the problem. Since the diagrams are generated outside the GA in advance, the GA cannot change or add new columns. The GA is therefore confined to finding only good combinations from a pre-determined pool of columns.

For the solution of CSP with a GA, chromosomes are normally represented by integer or binary vectors. Integer vector chromosomes contain only the numbers of the diagrams that constitute the schedule. This approach requires knowledge of the minimum number of diagrams in the schedule and this information is usually obtained from the lower bounds. Lower bounds are usually acquired through the solution of LP relaxation for SCP [13]. Since the number of diagrams tends to be higher than the lower bound, Costa et al [14] have suggested the following approach. In the first population the chromosomes have a length equal to the lower bound. Then, if a solution has not been found within a certain number of iterations, the length of the chromosome increases by one. This process repeats until the termination criteria are met.

In the binary vector representation, each gene stands for one diagram. The figure ‘1’ denotes that the diagram is included in the schedule, otherwise it is ‘0’. Such chromosomes usually consist of several hundred thousand genes, and only

around a hundred of them appear in the final solution. The number of diagrams can be unknown and the algorithm is likely to need a large number of iterations in order to solve the problem.

The application of genetic operators often violates the feasibility of the chromosomes, resulting in certain trips being highly over-covered (i.e. more than one driver assigned to the train) or under-covered (i.e. no drivers assigned to the train). One way of resolving this difficulty is to penalize the chromosome through the fitness function in accordance with the number of constraints that have been violated. However, the development of the penalty parameters can be problematic as in some cases it is impossible to verify them analytically and they are usually designed experimentally [15]. The penalty parameters are therefore data-dependent and likely to be inapplicable to other industries and companies. Moreover, the feasibility of the entire population is not guaranteed and might be achieved only after a large number of iterations.

Another more straightforward approach to maintaining the feasibility is to design heuristic "repair" operators. These operators are based on the principles "REMOVE" and "INSERT". They scan the schedule and remove certain drivers from the over-covered trips and assign those drivers to under-covered journeys [13, 15]. This procedure might have to be repeated several times, leading to high memory consumption and increased computation time.

# 3 GA-generated crew schedules

## 3.1 Initial data

The process starts with a user uploading the freight train and driver data (Figure 2). Each train has the following attributes: place of origin, destination, departure and arrival time, type of train and route code. The last two attributes indicate the knowledge that a driver must have in order to operate a particular train. The system also stores information about the drivers, i.e. where each driver is located and his or her traction and route knowledge. In the boxes marked 'traction knowledge' and 'route knowledge', each row represents a driver and each column denotes either a route or traction code. The binary digits indicate whether a particular driver is capable of driving a certain train or knows a certain route. The program also captures all the passenger trains and distance between cities, which is needed to calculate any taxi costs (Figure 3).

After all the necessary data have been uploaded, the GA is applied to construct an efficient schedule. The proposed algorithm overcomes the aforementioned challenges through a novel alternative chromosome representation and special decoding procedure. It allows the feasibility of chromosomes to be preserved at

each iteration without the application of repair operators. As a result, the computational burden is considerably reduced.

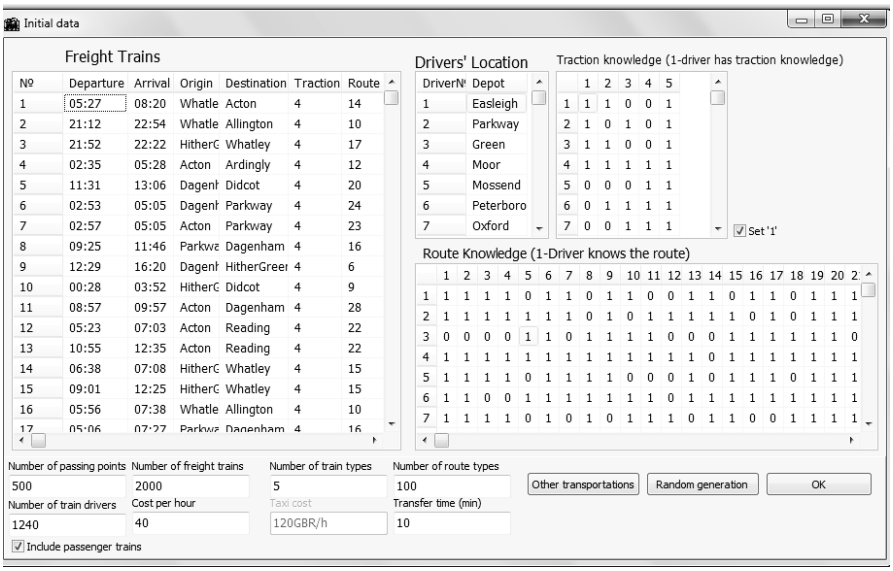


Figure 2 Freight trains and drivers

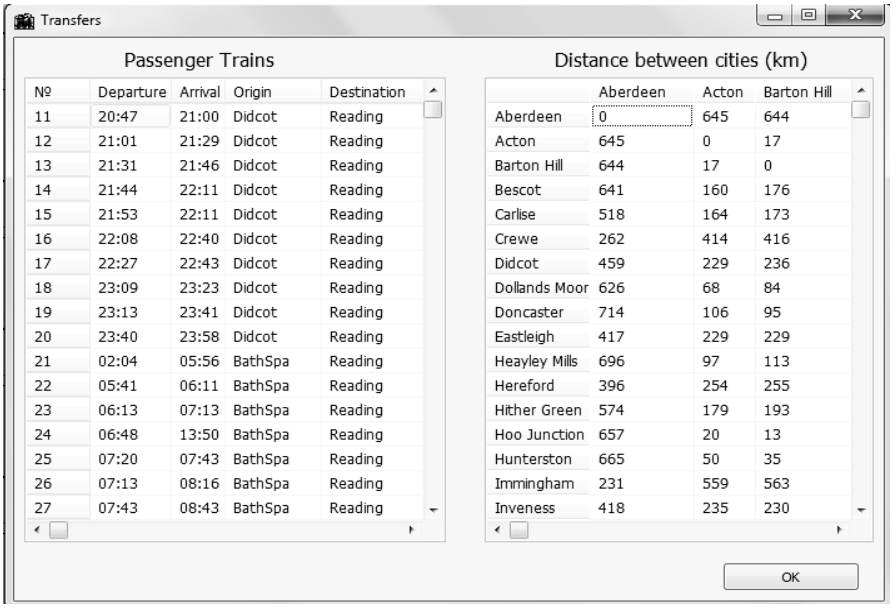


Figure 3 Passenger trains and taxis

## *3.2 Chromosome representation*

The chromosome is represented by a series of integers, where each number stands for the number of the trip (Figure 4). The population of chromosomes is generated at random and then the trips are allocated in series to the diagrams using a specific decoding procedure, which is discussed below.

Starting from the leftmost gene, the procedure finds a driver with the necessary route and traction knowledge to operate that trip and creates a new diagram for him or her. Then the procedure checks if the same driver is able to drive on the next journey (i.e. the second gene). If it is possible, then that trip is added to his or her diagram. If the origin station for the current trip differs from the destination station of the previous trip, the algorithm first searches for passenger trains and the freight company's own trains that can deliver a driver within the available time slot to the next job location, e.g. Diagram 1, between trips 3 and 8 (Figure 4). If no such trains have been found but there is a sufficient interval between the trips, then the algorithm inserts a taxi journey.

The information regarding driving times and the current duration of the diagrams is stored. Before adding a new trip, the algorithm inserts breaks if necessary. If the time expires and there are no trains to the home depot that a driver can drive, the deadheading activity completes the diagram, as in Diagram2 (Figure 4). If a trip cannot be placed in any of the existing diagrams, the procedure takes another driver from a database and creates a new diagram for him or her.

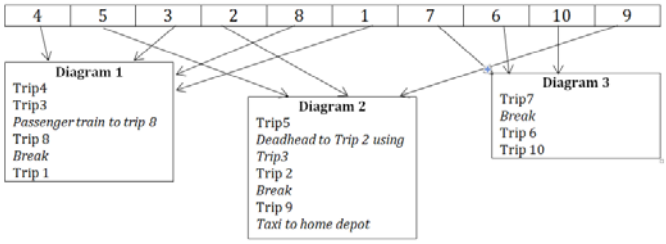


Figure 4 Chromosome representation and decoding procedure

On rare occasions, a few diagrams might be left with only a few trips and a duration that is less than the minimum. This is due to the fact that other drivers are either busy at this time or located at different stations. In order to tackle this problem, a mechanism has been added for finding and assigning a driver from a remote depot with the lowest workload. This approach not only solved the problem of the short diagrams, but also helped in distributing the workload more equally across the depots. After implementation of this procedure, the algorithm has been tested on various data sets including real and randomly generated data. Neither of the chromosomes has been reported to violate the constraint.

The given representation has a visual resemblance to the flight-graph representation suggested by Ozdemir and Mohan [16], but the decoding procedures are different. The flight-graph representation generates trips based on a depth-first graph search, whereas in the proposed GA they are produced at random. Random generation is beneficial since it does not exclude situations where a driver can travel to another part of the country to start working in order to have even workload distribution across the depots, while depth-first search usually places only geographically adjusted trips together.

The advantage of the proposed chromosome representation is that it creates both the diagrams and schedule within the same algorithm, thereby giving the GA greater control over the solution. It also does not require the generation of a large amount of diagrams at the beginning. In addition, this representation does not leave under-covered trips and ensures that no unnecessary over-covering happens. It is possible that at the beginning of the algorithm this chromosome representation might produce schedules with a high number of deadheads. However, due to the specific fitness function and genetic operators, the number of chromosomes containing deadheads decreases rapidly with evolution.

### *3.3 Fitness function*

An adequate solution of the CSP requires the achievement of two conflicting objectives: high throttle time and low deviation from average diagram lengths. It is evident that with the increase in throttle time, the deviation from the average diagram length will be increased towards a minimum diagram length. This is due to the algorithm attempting to allocate a diagram for a single trip in order to achieve 100% throttle time.

Since GAs are a single-objective optimization method, a weighted sum approach has been applied in order to transform all the objectives into scalar fitness information [17]. The advantage of this technique is relative simplicity of implementation as well as high computational efficacy [18].

### *3.4 Selection*

Preference was given to binary tournament selection as it is a comparatively simple and non-time consuming selection mechanism. It is also a popular selection strategy that is used in numerous GAs for CSP [9, 15, 16]. Binary tournament selection can be described as follows. Two individuals are selected at random form the population and the fittest among them constitutes the first parent. The same process repeats for the selection of the second parent.

### *3.5 Crossover and mutation*

Since one- or two-point crossover might produce invalid offspring by removing some trips or copying the same journey several times, a crossover mechanism utilizing domain-specific information has been designed. Firstly, the process detects genes responsible for diagrams with a high throttle time in the first parent. Then these genes are copied to the first child and the rest of the genes are added from the second parent. The same procedure is then used to form the second child. The process is illustrated on the Figure 5. By preserving the good parts of the chromosome accumulated through evolution, the implemented crossover was able to provide a schedule with a high throttle time much faster than traditional crossover that randomly the genes to propagate.


Figure 5 Crossover

In order to maintain diversity in the population, randomly selected genes are mutated with 40% probability. The mutation is performed by swapping two randomly identified genes. The mutation probability was determined through numerous tests and empirical observations.

## 4 Experimental results

The proposed GA for CSP (referred to as GACSP) has been used to produce diagrams for the freight-train drivers. The GACSP has been tested on a full daily data set obtained from one of the largest rail-freight operators in the UK. The data instances comprise 2000 freight-train legs, 500 cities, 39 depots, 1240 drivers, 500000 passenger-train links, and taxi trips connecting any of the stations at any time. Figures 6 and 7 illustrate a three-hour run of the algorithm and its achievement of the main business objectives, i.e. maximized throttle time and minimized deviation from the average shift duration. Increasing the throttle time indicates a reduction in deadheads and unnecessary waiting, thereby reducing the number of drivers required to operate the given trains. The decrease in deviation of the diagram duration from the average can be translated into equal utilization of the contract hours during the year. A typical resulting diagram is presented in

Figure 8, showing the sequence of trips and breaks that a driver needs to take on a particular day.

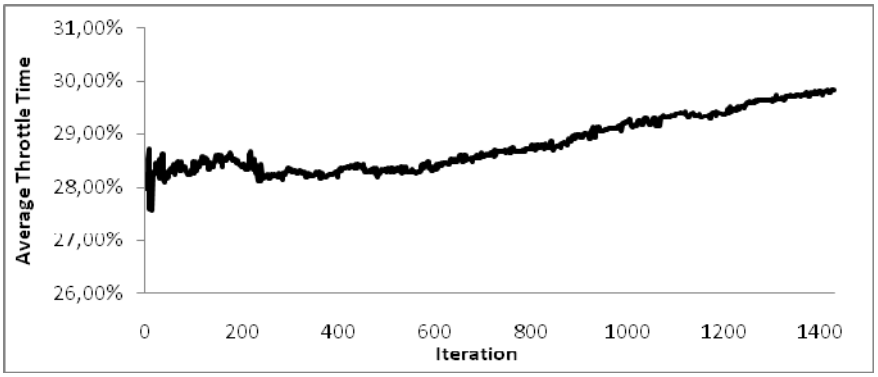


Figure 6 Maximizing average throttle time

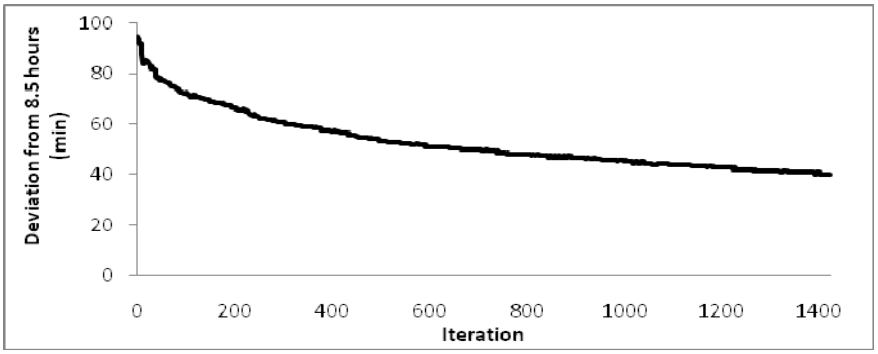


Figure 7 Minimizing deviation from the average shift length of 8.5 hours

| Driver | Start time | End time | Activity | Origin | Destination |
|---|---|---|---|---|---|
| 113 | 12:18 | 12:28 | Book on | Westbury | Westbury |
| 113 | 12:28 | 13:35 | Driving | Westbury | Swindon |
| 113 | 13:47 | 15:07 | Driving | Swindon | Wootton Wawen |
| 113 | 15:07 | 15:49 | Break | Wootton Wawen | Wootton Wawen |
| 113 | 15:59 | 17:29 | Driving | Wootton Wawen | Swindon |
| 113 | 17:37 | 21:00 | Passenger Train | Swindon | Westbury |
| 113 | 21:00 | 21:10 | Book off | Westbury | Westbury |
| **Diagram length: 8:52** | | | | **Throttle time: 46%** | |

Figure 8 A typical diagram, i.e. driver schedule

In order to evaluate the efficiency of GACSP, it has been compared against two established approaches. The first is B&P, i.e. the combination of column generation and branch and bound methods [4]. The second comparator is Genetic Algorithm Process Optimization (GAPO), a genetic algorithm for CSP enhanced with repair and perturbation operators [9]. Both GAs have been adapted and modified to the current problem and implemented with C++ Builder while B&P was written in CPLEX. They all were run on computer with 4 GB RAM and 3.4 GHz Dual Core Processor. Initially, the intention had been to test all three algorithms on the full data set. However, after twelve hours running of the B&P algorithm, no solution had been reached. For the sake of comparison, the data size was reduced to six cities and 180 train legs, 500 passenger-train links. For the GA the population size was set as 20, crossover rate 90% and mutation probability 40%. As criteria for comparison, real business objectives such as throttle time, number of deadheads, average deviation from the desirable diagram length and computation time have been selected.

Table 1 Experimental results using the reduced data set

| | B&P | | | GAPO | | | GACSP | | |
|---|---|---|---|---|---|---|---|---|---|
| Computation time (min) | 60 | 120 | 228 | 60 | 120 | 228 | 60 | 120 | 228 |
| Number of diagrams | - | - | 22 | 32 | 28 | 26 | 25 | 23 | 23 |
| Throttle time (%) | - | - | 63 | 50 | 56 | 59 | 60 | 62 | 62 |
| Average Number of deadheads per shift | - | - | 1.36 | 2.21 | 1.85 | 1.60 | 1.66 | 1.47 | 1.47 |
| Deviation from the average (min) | - | - | 46 | 51 | 48 | 47 | 62 | 57 | 57 |

The computational results with the reduced data sets are displayed in Table 1. B&P obtained a solution in 228 minutes. Within 10 minutes, B&P had constructed 2000 columns and solved LP relaxation without an integer solution. Further time was required for branching and generation of additional columns. In order to estimate efficiency of GACSP and GAPO, they have been run for the same period of time. GACSP obtained an entire feasible schedule within 10 seconds and after one hour an acceptable schedule had been reached. Although the B&P algorithm ultimately achieved slightly better results, it has been tested on a problem of relatively small size. The computational time for linear programming algorithms usually grows exponentially with the increase in data size, so the B&P algorithm is likely to be impractical in environments where there is a crucial need to make fast decisions from large data sets.

As in other work [9], 3308 columns have been generated for GAPO, which took 30 minutes of computational time. Unlike B&P and GACSP, this approach

did not have an embedded ability to generate additional columns, limiting its capability to explore other possible diagrams. It was also observed that 70% of the computational time was consumed by the heuristic and perturbation operators, whose aim was to restore the feasibility of the chromosomes. The repair operations were performed by scanning all available diagrams and selecting the best that could be inserted in the current schedule. GACSP overcomes this challenge by utilising the alternative chromosome representation that does not violate the validity of the chromosomes. Thus GACSP spends less time on each iteration and hence evolves more rapidly.

## 5 Potential implementation and integration issues

The most common implementation problems with software for scheduling transit systems concern robustness [19], i.e. the ability of the schedule to adapt to different circumstances. An example of such circumstances might be the delay of the previous train, resulting in the driver being unable to catch the planned train. In our system, the transfer time regulates how much time is allocated for a driver to leave the previous train and start working on the next one. The larger the interval between trips, the lower the risk that the next freight train will be delayed by the late arrival of the previous one. On the other hand, a large transfer time decreases throttle time and requires more drivers to cover the trips. The best way to tackle this situation is to have an effective re-scheduling mechanism that makes changes in as few diagrams as possible.

In addition, the crew scheduling process is extremely complex. It is not always possible to model all the rules, nuances and exceptions of the schedule. For this reason, the system-generated diagrams have to be revised and amended by an experienced human planner until all the knowledge has been fully acquired.

Finally, although GAs are able to find an acceptable solution relatively quickly, they might also converge prematurely around a sub-optimal solution. Convergence can be controlled either by embedding variations in the selection procedure [17] or by changing the mutation rate [13].

## 6 Conclusions

In this paper, the complexities of CSP in the rail-freight industry in the UK have been described. Due to a high monetary cost of train crew, the profitability and success of the company might rely heavily on the quality of the constructed crew schedule. Given the wide geographical spread, numerous regulations, and severely constrained planning time, an IT system with an effective scheduling algorithm can equip a company with valuable decision-making support.

We have proposed a novel GA for crew scheduling (GACSP). Unlike other GAs for CSP, GACSP works with the entire schedule and does not restrict the algorithm in finding an optimal solution. The special chromosome representation and genetic operators are able to preserve the validity of the chromosomes without the need for additional repair operators or penalty functions. This capability enables the algorithm to consume fewer memory resources and to find a solution faster. In addition, the user can to retrieve a feasible schedule at any iteration.

It has been shown that although B&P was capable of finding an optimal solution from the mathematical perspective, time was its main weakness. In real-world operations, the cost for late optimal decision often can be much higher than that of a fast sub-optimal one. In this sense, the GA demonstrated excellent results as it provided a reasonable schedule nearly four times faster when using the reduced rail network. When faced with the scheduling task for the complete UK rail network, B&P had failed to find a solution at all after 12 hours, whereas GACSP was able to find an adequate solution in 2 hours. With further improvements of GACSP and possible hybridization with linear programming methods, its performance maybe further improved.

As future work, more domain specific rules will be incorporated into the chromosome generation process in order to achieve a better initial population. Moreover, it would be worthwhile to investigate a possible hybridization of a GA with the B&P method. The hybridization might seize the advantages of both algorithms to reach a solution that is close to the mathematical optimum in a short computation time.

## Acknowledgements

This research has been supported by Sheffield Business School at Sheffield Hallam University. The authors would like to thank DB-Schenker Rail (UK) for the provision of data and information about real-world crew scheduling operations.

## References


1. Barnhart, C., Hatay, L., Johnson, E.L.: Deadhead selection for the long-haul crew pairing problem. Oper. Res. Vol.43(3), pp.491-499 (1995).
2. Drexl, M., Prescott-Gagnon., E: Labelling algorithms for the elementary shortest path problem with resource constraints considering EU drivers' rules. Log. Res. Vol. 2(2), pp. 79-96 (2010).
3. Duck, V., Wesselmann, F., Suhl, L.: Implementing a branch and price and cut method for the airline crew pairing optimization problem. Pub. Transp..Vol.3(1), pp.43-64 (2011).
4. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W., Vance, P.H.: Branch-and-price: Column generation for solving huge integer programs. Oper. Res. Vol.46(3), pp.316-329 (1998).
5. Derigs, U., Malcherek, D., Schafer, S.: Supporting strategic crew management at passenger railway model, method and system. Pub. Transp. Vol.2(4), pp.307-334 (2010).
6. Niederer, M.: Optimization of Swissair's Crew Scheduling by Heuristic Methods Using Integer Linear Programming Models. AGIFORS Symposium (1966).
7. Levine, D.:Application of a hybrid genetic algorithm to airline crew scheduling. Comp. & Oper. Res. Vol. 23(6), pp.547-558 (1996).
8. Santos, A.G., Mateus, G.R.: General hybrid column generation algorithm for crew scheduling problems using genetic algorithm. Proceedings of the 2009 Congress on Evolutionary Computation. pp. 1799-1806 (2009).
9. Zeren, B., Özkol, I.: An improved genetic algorithm for crew pairing optimization. J. Intell. Learn. Sys. & Appl. Vol.4(1), pp. 70-80 (2012).
10. Souai, N., Teghem, J.: Genetic algorithm based approach for the integrated airline crew-pairing and rostering problem. Eur. J. Oper Res. Vol. 199(3), pp. 674-683 (2009).
11. Park, T., Ryu, K.. :Crew pairing optimization by a genetic algorithm with unexpressed genes. J. Intell. Manuf. Vol. 17(4), pp. 375-383 (2006).
12. Kornilakis, H., Stamatopoulos, P.: Crew pairing optimization with genetic algorithms. In: Methods and Applications of Artificial Intelligence, pp. 109-20, Springer (2002).
13. Kwan, R.K., Wren, A., Kwan, A.K.: Hybrid genetic algorithms for scheduling bus and train drivers. Proceedings of the 2000 Congress on Evolutionary Computation.Vol.1, pp.282-292 (2000).
14. Costa, L., Santo, I.E., Oliveira, P.: An adaptive constraint handling technique for evolutionary algorithms. Optimization. Vol.62(2), pp. 241-253 (2013).
15. Chu, P.C., Beasley, J.E.: Constraint handling in genetic algorithms: The set partitioning problem. J. Heuristics. Vol. 4(4), pp.323-57 (1998)
16. Ozdemir, H.T., Mohan, C.K.: Flight graph based genetic algorithm for crew scheduling in airlines. Inf Sci.Vol.133(3-4), pp. 165-173 (2001).
17. Hopgood, A.A.: Intelligent systems for engineers and scientists. 3rd ed. CRC Press, Boca Raton (2012).
18. Coello, A.C.: An updated survey of GA-based multiobjective optimization techniques. ACM. doi:10.1145/358923.358929 (2000).
19. Gopalakrishnan, B., Johnson, E.L.: Airline crew scheduling: State-of-the-art. Ann. of Oper.Res. Vol. 140, pp. 305-337 (2005).