A Genetic Algorithm Approach for Scheduling Trains Maintenance under Uncertainty

Hanyu Gu and Hue Chi Lam*

School of Mathematical and Physical Sciences, University of Technology Sydney, Sydney, NSW, Australia.
hanyu.gu@uts.edu.au (Hanyu Gu)
hue.lam@student.uts.edu.au (Hue Chi Lam)

Abstract. This paper investigates the overhaul maintenance scheduling problem in which the maintenance duration is uncertain at the time of planning. This problem involves specifying the dates of trains' arrival at the maintenance centre while taking into consideration the due windows, the desired number of trains in service, and the capacity of the maintenance centre. The cycle time of each type of trains is random with a known probability distribution. The objective is to minimise a weighted sum of two components: (i) the deviation of the assigned arrival dates from the due windows and (ii) the penalty for violating the resources' constraints. A combined genetic algorithm with sample average approximation solution approach is developed to solve this problem. The solution approach consists of a genetic algorithm for global search and an exact method to determine the arrival dates of train-sets when a sequence of train-sets is known. The results with data provided by one of the leading Australian maintenance center show that the proposed method can produce good solution within acceptable computation time.

Keywords: Genetic algorithm · Stochastic cycle time· Quadratic earliness/tardiness · Sample average approximation.

1 Introduction

This paper deals with the scheduling of overhaul maintenance of trains, arising in the realm of passenger rail services. In the case of overhaul maintenance, the trains are withdrawn from service and are sent to a specialised maintenance centre where they will stay for at least one month for the entire maintenance process. A typical objective of this problem is to minimise the total cost of earliness and tardiness subject to the centre capacity. According to [8], this problem is known as the Overhaul Maintenance Scheduling Problem (OMSP). This paper addresses the stochastic version of OMSP, where the dwell time of the trains at the maintenance centre is uncertain at the time of planning. We refer to this problem as Stochastic Overhaul Maintenance Scheduling Problem (SOMSP).

* Corresponding author

In SOMSP, we consider the perspective of the maintenance centre's planning manager who must determine an arrival plan specifying the arrival dates for all trains for one year or for a longer period, where each train has a distinct due window and a random dwell time[1] that follows a known probability distribution associated with the type of train. During the planning phase, one must take into consideration the distinct due windows which are the desired arrival windows of the trains. It is noted that the definition of due window in this paper is different from those in most scheduling literature whereby it is referred to as the desired completion window of a job.

The train arrives at the maintenance centre in groups. Each group is comprised of several cars coupled together and is referred to as a set or a train-set (see for example [4]). All cars in a train-set undergo the maintenance in the maintenance centre simultaneously. A feasible arrival plan is one that satisfies a number of constraints. Firstly, a team of technicians and engineers with a broad range of skills is needed to perform the maintenance tasks. To complete the maintenance operations, the team must use various equipment and materials. These renewable and nonrenewable resources can be quantified as centre capacity. The centre capacity imposes a restriction on the number of train-sets which can undergo maintenance simultaneously.

Secondly, on the arrival date, the train-set is completely withdrawn from service and must stay at the maintenance centre for at least one month. This long cycle time directly impacts the number of train-sets available in active service. Therefore, a permissible number of train-sets that can be taken out of service simultaneously are specified for each type of train-sets.

Solving the SOMSP is challenging for various reasons: (i) the single machine scheduling problem with earliness and tardiness objectives, which is similar to the OMSP, is known to be NP-hard [11]. Hence, the considered SOMSP with stochastic dwell time is even harder to solve; and (ii) due to the non-uniform distribution of the desired arrival window, there is a trade-off between respecting the time window and satisfying the centre capacity and operational requirement [6].

Due to the complexity of SOMSP, a Genetic Algorithm (GA) is proposed to solve the considered problem. Based on previous study in [2], it is noted that the decoding procedure, where a chromosome is transformed into a feasible arrival plan, is a crucial step of GA. The decoding procedure used in [2] is a simple greedy heuristic which does not generate good solution. To improve the performance, we develop a new decoding procedure based on Sample Average Approximation (SAA) method in this paper.

The remainder of this paper is organised as follows. Section 2 presents the relevant literature. Section 3 gives the mathematical formulation of the considered problem. Section 4 shows the model formulation using SAA approach. Section 5 describes the proposed genetic algorithm in detail. Section 6 reports the results

[1] In the context of this paper, dwell time and cycle time have the same meaning. The two terms are used interchangeably.

of the computational experiments. Finally, section 7 concludes the study and gives directions for future research.

2 Literature review

Papers that consider the scheduling and planning of rolling stock's maintenance within the railway industry can be categorised into two groups. The first focuses on the running maintenance which is performed during the connection between two consecutive paths or overnight at the rolling stock depot. These studies generally consider the perspective of the railway provider who must design a rolling stock rostering plan while taking into consideration various maintenance requirements. The design objectives often include minimising a weighted sum of maintenance cost, deadhead cost due to unexpected failure, and substitution cost due to the use of undesired train-sets [4]; and minimising the sum of deadhead costs and number of train units [9].

The second focuses on the overhaul maintenance which is performed at a specialised workshop. These studies generally consider the perspective of the workshop's planning manager who must determine the dates on which the trains must be sent to the workshop for maintenance. As an early work, [8] proposes a genetic algorithm for solving the OMSP. Doganay & Bohlin [1] formulates the OMSP as a mixed integer linear programming model and solves it by exact method. Lin et al. [6] formulates the high-level maintenance scheduling for high speed trains as a mixed integer linear programming model and proposes a simulated annealing algorithm for solving large-scale instances. From these studies, we can observe that the objective of minimising the total cost of earliness and tardiness is a common design objective of OMSP.

The proposed SOMSP is similar to a Stochastic Multiple Resource Constrained Scheduling Problem (SMRCSP), in which one need to decide the start times of jobs requiring different types of resources under uncertain processing time [3]. This kind of problem has application in appointment sequencing and scheduling where jobs correspond to operation appointments and resources correspond to doctors, nurses and operating rooms [7]. However, the key differences between the proposed SOMSP and the aforementioned problems involve: (i) each type of train-sets has a known processing time on the first operation line where preemptions are not allowed. That is, the scheduling on the first operation line can be treated as a single machine scheduling problem; and (ii) for train-sets of the same type, there exists a noticeable sequence among them. That is, train-set with earlier due window should always arrive for maintenance earlier than the others of the same type in order to minimise the objective function value.

3 Mathematical formulation

Consider a set $N := \{1, \cdots, n\}$ of train-sets and a planning period of T days. The planning horizon is discretised into calendar days which are indexed from 0 to $T - 1$.

Each train-set $j \in N$ has a due window $[e_j, l_j]$, where e_j is the earliest desired arrival date and l_j is the latest desired arrival date. The earliness and tardiness of train-set j if it arrives on day t are defined as $E_{jt} = \max\{0, e_j - t\}$, and $T_{jt} = \max\{0, t - l_j\}$, respectively. Let c_{jt} be the cost for assigning train-set j to arrive on day t, as indicated in (1).

$$c_{jt} = \lambda_1 E_{jt}^2 + \lambda_2 T_{jt}^2, \tag{1}$$

where λ_1 and λ_2 are the earliness and tardiness cost factors, respectively.

There are m types of train-sets and the set of all train-sets is partitioned into m families. Each train-set belongs to a train family $F^k, k \in K = \{1, \cdots, m\}$. Each type of train-sets requires a minimum duration on the first operation line, during which the maintenance operations must be performed without interruption. For each train family F^k, let p_k be the minimum duration on the first operation line. The minimum duration is assumed to be deterministic.

For each day t, let C_t be the centre capacity (i.e. the number of train-sets which can undergo maintenance simultaneously). Violation of this restriction is permitted for a penalty cost. Let δ be the unit penalty for violating the centre capacity. In practice, the penalty is associated with each additional train-set to account for overtime, outsourcing, and hiring contractors.

For each train family F^k and each day t, let C_{kt} be the permissible number of out-of-service train-sets. Violation of this restriction is permitted since the shortage of some types of train-sets can be substituted by a different train type. Let δ_{kt} be the unit penalty for violating the limit C_{kt}. In practice, a penalty is associated with each substitution to account for the passengers' dissatisfaction due to the differences in their configurations.

The cycle time of each train-set j is a random variable D_j. Train-sets in a train family follows the same probability distribution. It is assumed that the probability distribution for the cycle time of each type of train-sets is known. It is further assumed that all D_j are independent.

We introduce the time indexed binary variables $x_{jt} \in \{0, 1\}$ which is equal to 1 if train-set $j \in N$ arrives at the maintenance centre on day t, and is equal to 0 otherwise. Then, for each train-set $j \in N$ its arrival day is defined as

$$s_j = \sum_{t=0}^{T-1} t x_{jt}, \ \forall j \in N \tag{2}$$

Accordingly, we can define an arrival plan $s = (s_1, \cdots, s_n)$. For any arrival plan s and any integers $1 \leq k \leq m$ and $0 \leq t < T$, the number of trains of family k that present at the centre on day t is

$$Z_{kt}^s = \sum_{j \in F^k} B(s_j, t), \tag{3}$$

where

$$B(s_j, t) = \begin{cases} 1 \text{ if } s_j \leq t \text{ and } s_j + D_j \geq t + 1 \\ 0 \text{ otherwise} \end{cases}. \tag{4}$$

Then, the total number of trains present at the centre on day t is

$$Z_t^s = \sum_{k \in K} Z_{kt}^s. \tag{5}$$

If it is clear which arrival plan is considered, the superscript s can be dropped and the notation Z_t (Z_{kt}) can be used instead of Z_t^s (Z_{kt}^s).

Formulation of SOMSP

$$\begin{aligned} \text{Min } & \alpha \sum_{j \in N} \sum_{t=0}^{T-1} c_{jt} x_{jt} \\ & + \beta \sum_{t=0}^{T-1} \left(\delta \mathbb{E}\left[(Z_t - C_t)^+ \right] + \sum_{k \in K} \delta_{kt} \mathbb{E}\left[(Z_{kt} - C_{kt})^+ \right] \right) \end{aligned} \tag{6}$$

subject to

$$\sum_{t=0}^{T-1} x_{jt} = 1, \quad \forall j \in N \tag{7}$$

$$\sum_{k \in K} \sum_{j \in F^k} \sum_{s=\max(t-p_k+1,0)}^{t} x_{js} \leq 1, \quad \forall t \in [0, T-1] \tag{8}$$

(3), (4), (5)

$$x_{jt} \in \{0,1\}, \quad \forall j \in N, \quad \forall t \in [0, T-1] \tag{9}$$

Where $(a)^+ := \max(a, 0)$; $E[.]$ denotes the expectation operator; α and β are weights reflecting the relative importance of the two components of the objective function. The objective function (6) minimises the weighted sum of two components: the cost incurred if train-set j arrives for maintenance on day t (i.e. the quadratic earliness and tardiness costs); and the expected penalties for violating the limits C_t and C_{kt}. Constraint (7) guarantees that each train-set is scheduled for maintenance on a particular day t within the planning horizon. Constraint (8) depicts the restriction on the first operation line. If a train-set j occupies the first operation line in a given time interval, other train-sets are not allowed to arrive during this period. Constraint (9) states that the decision variables are binary.

4 SAA Model

Using the sample average approximation approach, the SOMSP formulation, presented in section 3, can be rewritten as a mixed integer programming model.

Consider a set $\Omega := \{1, \cdots, \omega\}$ of scenarios. Each scenario comprises a vector of realisations of cycle time which are drawn independently from the distributions corresponding to each train family. Let ξ_j^ω be the cycle time of train-set j in scenario ω.

Formulation of SAA Model

$$\text{(SAA) Min } \alpha \sum_{j \in N} \sum_{t=0}^{T-1} c_{jt} x_{jt} + \beta \, \frac{1}{\mid \Omega \mid} \sum_{\omega \in \Omega} \left(\sum_{t=0}^{T-1} \left[\delta z_t^{\omega} + \sum_{k \in K} \delta_{kt} z_{kt}^{\omega} \right] \right) \tag{10}$$

subject to

(7), (8), (9)

$$\sum_{j \in N} \sum_{s=\max(t-\xi_j^{\omega}+1,0)}^{t} x_{js} \leq C_t + z_t^{\omega}, \qquad \forall \omega \in \Omega, \quad \forall t \in [0, T-1] \tag{11}$$

$$\sum_{j \in F^k} \sum_{s=\max(t-\xi_j^{\omega}+1,0)}^{t} x_{js} \leq C_{kt} + z_{kt}^{\omega}, \qquad \forall \omega \in \Omega, \quad \forall k \in K, \quad \forall t \in [0, T-1] \tag{12}$$

$$z_t^{\omega} \geq 0, z_{kt}^{\omega} \geq 0, \qquad \forall \omega \in \Omega, \quad \forall k \in K, \quad \forall t \in [0, T-1] \tag{13}$$

The objective function (10) is the weighted sum of two components: the first component is the same as the first component of objective function (6), while the second component is the sample average of the penalties for the additional train-sets. We call the first component and second component of objective function (10), the *earliness tardiness cost* (ETC) and the *resource violation cost* (RVC), respectively. For every scenario ω, constraints (11) and (12) describe the additional train-sets, represented by the slack variables z_t^{w} (z_{kt}^{w}), based on the limits C_t and C_{kt}. Constraint (13) is the non-negativity constraint.

4.1 Property of SAA Model

SAA has been extensively used in literature to solve stochastic optimisation problems (see for example, [7]). However, solving SAA model by itself can be computationally challenging as the number of scenarios increases. To demonstrate the complexity of SAA model, we perform a preliminary experiment with various number of scenarios. The result in term of computation time is reported in Table 1. In the same table, we also present the computation time required for solving SAA model if a sequence is given. Given a set of train-sets, a sequence is the order in which the train-set arrives at the maintenance centre. Let E be a set of arcs representing the precedence relations obtained from the sequence. The SAA model with sequence (SAA-sequence) includes the addition of the following constraint:

$$\sum_{t=0}^{T-1} t \; x_{it} \leq \sum_{t=0}^{T-1} t \; x_{jt}, \quad \forall (i,j) \in E \tag{14}$$

Constraint (14) enforces the precedence relations among train-sets according to the given sequence.

Table 1. Computation time (in seconds) required for solving SAA model with and without sequence

Number of scenarios	SAA model	SAA-sequence model
5	288.348	9.375
10	-	21.146
20	-	44.947
30	-	62.048
40	-	81.929
50	-	129.034
100	-	421.410

From Table 1, we note that SAA model cannot be solved within the time limit of one hour when the number of scenarios is larger than or equal to 10 scenarios. On the other hand, solving SAA model with given sequence is relatively easier. Problem instance with 100 scenarios can be solved in less than 500 seconds.

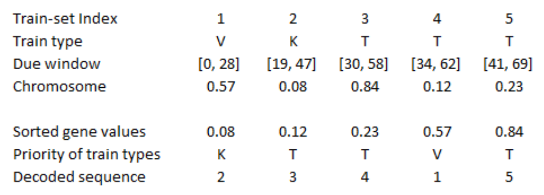
5 The solution method

Motivated by the result in section 4, a genetic search with SAA (GS-SAA) is proposed to solve the problem. The idea of GS-SAA is simple. It combines genetic algorithm for global search and solving SAA-sequence model for a solution. The genetic algorithm is inspired by [5] and has previously been studied in [2]. Detail of the proposed GS-SAA is given below.

Algorithm

```
 1: Generate initial population
 2: Decode the chromosomes into solutions by solving SAA-sequence model
 3: Evaluate the solutions
 4: for the number of generations < $IT_{max}$, and time < $T_{max}$ do
 5:     Select parents
 6:     Generate offspring (two-point crossover)
 7:     Diversify offspring (mutation)
 8:     Decode offspring into solutions by solving SAA-sequence model
 9:     Evaluate the solutions
10: end for
11: return the best solution
```

The proposed GS-SAA starts with the generation of an initial population of size μ. Then, each chromosome in the initial population is decoded into a sequence. The sequence decoding procedure works as follow. An arrival plan, or a solution P is represented by a chromosome. Each gene in the chromosome corresponds to a train-set and the gene value is generated randomly from the uniform distribution $U(0, 1)$. The gene values determine the priority of the train types. Respecting this priority list, the priority of train-sets (i.e. the sequence of train-sets) is obtained by sorting the train-sets in the same train family based on its earliest desired arrival date (see Figure 1 for an example).



Train-set Index	1	2	3	4	5
Train type	V	K	T	T	T
Due window	[0, 28]	[19, 47]	[30, 58]	[34, 62]	[41, 69]
Chromosome	0.57	0.08	0.84	0.12	0.23
Sorted gene values	0.08	0.12	0.23	0.57	0.84
Priority of train types	K	T	T	V	T
Decoded sequence	2	3	4	1	5

Fig. 1. Example of sequence decoding by train types.

Once the sequence is known, an arrival plan, or a solution P is generated by solving the SAA-sequence model. Next, the solution P is evaluated according to the objective function (6). The evaluation is the same as that given by [2].

Given the current generation, the next generation is produced through elite selection, crossover, and mutation.

In the elite selection phase, some of the best individuals of the current generation will continue to exist in the next generation. The number of surviving individuals is equal to N_{elite}. The elite selection strategy is motivated by the "Survival of the fittest" [10]. However, the drawback is that it can lead to premature convergence (i.e. catch in a local optimum).

In the crossover phase, a child is produced based on the two-point crossover as in [2]. The two parents are randomly selected from a pool of parents. This pool consists of the top N_{elite} best individuals of the current generation and the remaining candidates are created from the Roulette Wheel Selection. Each proportion of the wheel is given to an individual of the current generation. The size of the proportion is equivalent to the probability of selection, which is defined as

$$\text{Probability of selection} = \frac{\text{fitness}}{\text{sum of fitness}},$$

where fitness is equal to the inverse of the value of the objective function (6), and sum of fitness is equal to the total fitness of all individuals.

In the mutation phase, some gene values of the child chromosome will be replaced by a random number sampled from the uniform distribution $U(0, 1)$. The replacement is initiated when the random number is less than P_m. Detail of the mutation process is discussed in [2].

6 Computational results

The GS-SAA method was coded in Python 2 on an Intel Core i5-6500 CPU @3.2 Ghz with 8GB of RAM. The SAA-sequence problem was solved using the Branch-and-Cut method in IBM ILOG CPLEX 12.6.1 on the same computer.

We adopt the test problems in [2] where there are 3 train families with 35 train-sets in total. The planning horizon is set to one year. The parameters associated with the train families are shown in Table 2. Furthermore, we set $C_t = 5$, $\delta_t = 1$, $\lambda_1 = \lambda_2 = 1$, $\alpha = 1$, and $\beta = 1000$. For the setting of our genetic algorithm, we have $\mu = 20$, $N_{elite} = 2$, $P_m = 0.05$, $IT_{max} = 40$, and $T_{max} = 3600$ (seconds). The information on the probability distributions of the cycle time by train family is given in Table 3.

Table 2. Parameters for the train families

Train Family	$\lvert F^k \rvert$	p_k	Norm		Special	
			C_{kt}	δ_{kt}	C_{kt}	δ_{kt}
1	25	4	3	1	1	10
2	5	5	2	1	1	10
3	5	5	1	1	1	10

Norm refers to normal day and Special refers to special days.

Table 3. Probability distribution information for cycle time by train family.

Train Family	Minimum	Most likely	Maximum	Distribution
1	20	25	40	beta-PERT
2	27	30	46	beta-PERT
3	29	30	52	beta-PERT

In using SAA, one question to ask is how many scenarios are required in order to obtain solution with good quality within acceptable computation time. To answer this question, we extend the preliminary experiment in section 4.1 for the SAA-sequence model such that both the solution quality and computation time are presented. The SAA-sequence model is used to solve the test problems with 5, 10, 20, 30, 40, 50, and 100 scenarios. The scenarios are randomly generated. The average objective value and average computation time obtained after 5 runs are reported in Table 4.

Table 4. Average results for 5 runs of SAA-sequence for different number of scenarios

Number of scenarios	Computation time (s)	Objective Value	ETC	RVC
5	9.375	206,760	22,447	184.318
10	21.146	205,086	22,613	182.473
20	44.947	199,781	22,743	177.038
30	62.048	199,046	22,743	176.030
40	81.929	199,046	22,743	176.030
50	129.034	199,046	22,743	176.030
100	421.410	199,046	22,743	176.030

From Table 4, it can be seen that the computation time increases significantly as the number of scenarios increases. As a good trade-off between solution quality and computation time, SAA-sequence with up to 10 scenarios is sufficient. Therefore, we set $|\Omega| = 1, 5, 10$.

Table 5 shows the performance of our GS-SAA method with 1, 5, and 10 scenarios, in comparison with the method proposed in [2] (denoted as MIM) and solving SAA model by itself with 5 scenarios (denoted as SAA-5). For GS-SAA and MIM solution aprroaches, the best solution of the initial population is reported under the column titled “In.”, while the best solution of the final population is reported under the column titled “obj.”. For SAA-5 solution approach, the optimal solution obtained by CPLEX is reported under the column titled “CPLEX obj.”. For each solution approach, 5 runs of experiment are performed, the average, maximum and minimum of the objective values are also shown in Table 5.

On average, GS_SAA with 10 scenarios produces better solution for the initial population. This result is not surprising since the solution quality of the approximation of the objective function (6) by its sample average increases with increasing number of scenarios as demonstrated in Table 4.

To test the impact of the number of scenarios on the performance of GS_SAA, we consider the relative improvement in the objective function value over the best solution of the initial population and it can be calculated by $(In. - obj.)/In. \times 100\%$. On average, the relative improvement in the objective function value over the best solution of the initial population is approximately 27.5%, 28.8%, 19.9%, respectively, for 1, 5, 10 scenarios. The small relative improvement of GS_SAA with 10 scenarios is due to the amount of time required for solving SAA-sequence model; as a result, only a few generations are explored and the search capability of genetic algorithm is not fully employed. This observation suggests that GS_SAA with 10 scenarios can potentially give better solution if it is allowed to run for a longer period of time.

Furthermore, as shown in Table 5, the performance of GS_SAA is consistently better than MIM irrespective of the number of scenarios used. The method used in MIM to transform a sequence into a feasible arrival plan is just a simple greedy heuristic and it suffers from poor performance as the idle time inserted between train-sets is not optimal. On the other hand, given the sequence, GS_SAA generates an optimal arrival plan by solving the corresponding SAA-sequence model using CPLEX.

Compare GS_SAA with SAA-5, it can be noted that the latter outperforms GS_SAA in all runs of experiment. This reveals that a good sequence is crucial to obtaining a good solution. However, as demonstrated in Table 1, solving SAA model by itself is computationally challenging if the number of scenarios increases or the problem size becomes large (CPLEX cannot obtain optimal solution in 1 hour given the problem size in this paper). This shows the limitation of SAA model in large applications.

Table 5. Comparison of the performance of the different solution approaches (solution quality).

Run	MIM		GS_SAA						SAA-5
			1 scenario		5 scenarios		10 scenarios		
	In.	obj.	In.	obj.	In.	obj.	In.	obj.	CPLEX obj.
1	607,466	475,601	343,337	248,974	290,096	247,636	331,497	230,787	218,949
2	533,746	475,601	340,894	227,894	344,851	238,421	330,219	246,854	226,197
3	624,286	475,601	379,608	242,879	370,441	238,200	314,281	280,038	213,755
4	629,515	475,601	344,171	331,267	338,932	234,148	331,583	282,617	212,428
5	593,304	475,601	320,719	300,493	345,213	243,890	348,886	286,488	216,267
Average	597,663	475,601	345,745	270,301	337,906	240,495	331,293	265,357	217,519
Max	629,515	475,601	379,607	331,267	370,441	247,636	348,886	286,488	226,197
Min	533,746	475,601	320,719	227,894	290,096	234,148	314,281	230,787	212,428

In. is the best solution of the initial population.
obj. is the objective value. CPLEX obj. is the optimal solution obtained by CPLEX

Table 6 shows the computation times in seconds obtained by MIM, GS_SAA with 1, 5 and 10 scenarios, and SAA-5. On average, SAA-5 can be solved to optimality in 288.35 seconds, whereas MIM takes more than 1288.55 seconds, and GS_SAA takes 1972.81, 3844.88, 4137.58 seconds, respectively, for 1, 5, 10 scenarios. Both MIM and GS_SAA require significant computation time because both methods could only terminate after the predefined maximum number of generations and time limit are reached.

Table 6. Comparison of the performance of the different solution approaches (Computation time).

Run	MIM	GS_SAA			SAA-5
		1 scenario	5 scenarios	10 scenarios	
1	1263.52	2053.31	3796.17	4132.10	137.55
2	1255.65	2036.93	3921.30	4104.96	202.65
3	1439.97	1903.90	3803.61	4051.59	135.22
4	888.78	1881.32	3829.91	4097.07	254.10
5	1594.85	1988.57	3873.41	4300.58	712.23
Average	1288.55	1972.81	3844.88	4137.58	288.35
Max	1594.85	2053.31	3921.30	4300.58	712.23
Min	888.78	1881.32	3796.18	4051.59	135.22

7 Conclusions

This paper examines the overhaul maintenance scheduling problem with stochastic cycle time. We present the mathematical formulation of SOMSP and show how it can be formulated as a mixed integer programming model using the sample average approximation approach. A combined genetic search with SAA is developed to solve the problem. The result shows that solving the SAA model by itself is computationally challenging as the number of scenarios become large.

However, if a sequence of train-sets is given, the computation time decreases significantly. Computational results reveal that the proposed GS-SAA can produce good solution within acceptable time for test problem consisting of 35 train-sets and a planning horizon of one year.

In conclusion, SAA can be used for small instances since CPLEX can obtain optimal solution within acceptable computation time. For larger application, GS_SAA is the preferred choice. It is noted that the performance of GS_SAA depends on having both a good sequence and good inserted idle time. Future work can investigate methods to further reduce the computation time of SAA-sequence model to enable the search capability of the genetic algorithm.

References

1. Doganay, K. & Bohlin, M. (2010). Maintenance Plan Optimization for a Train Fleet. *WIT Transactions on the Built Environment*, **114**, 349-358.
2. Gu, H., Joyce, M., Lam, H. C., Woods, M., & Zinder, Y. (2019) A Genetic Algorithm for Assigning Train Arrival Dates at a Maintenance Centre. *9th IFAC Conference on Manufacturing Modelling, Management and Control (MIM), Berlin, Germany, 28-30 August 2019.*
3. Keller, B., & Bayraksan, G. (2010). Scheduling jobs sharing multiples resources under uncertainty: A stochastic programming approach. *IIE Transactions*, **42**, 16–30.
4. Lai, Y. C., Fan, D. C., & Huang, K. L. (2015). Optimizing rolling stock assignment and maintenance plan for passenger railway operations. *Computers & Industrial Engineering*, **85**, 284–295.
5. Li, H., & Demeulemeester, E.(2016). A genetic algorithm for the robust resource leveling problem. *Journal of Scheduling*, **19**(1), 43–60.
6. Lin, B., Wu, J., Lin, R., Wang, J., Wang, H., & Zhang, X. (2019). Optimization of high-level preventive maintenance scheduling for high-speed trains. *Reliability Engineering & System Safety*, **183**, 261-275.
7. Mancilla, C. & Storer, R.(2012). A sample average approximation approach to stochastic appointment sequencing and scheduling. *IIE Transactions*, **44**(8), 655-670.
8. Sriskandarajah, C., Jardine, A. K. S., & Chan, C. K.(1998). Maintenance Scheduling of Rolling Stock Using a Genetic Algorithm. *The Journal of the Operational Research Society*, **49**(11), 1130-1145.
9. Tsuji, Y., Kuroda, M., Kitagawa, Y., & Imoto, Y. (2012) Ant Colony Optimization approach for solving rolling stock planning for passenger trains. *2012 IEEE/SICE International Symposium on System Integration (SII), Fukuoka, Japan, 16-18 December 2012.*
10. Valente, J. M. S., Moreira, M. R. A., Singh, A., & Alves, R. A. F. S. (2011). Genetic Algorithms for Single Machine Scheduling with Quadratic Earliness and Tardiness Costs. *The International Journal of Advanced Manufacturing Technology*, **54**(1), 251-265.
11. Wan, L. & Yuan, J.(2013). Single-machine scheduling to minimize the total earliness and tardiness is strongly NP-hard. *Operations Research Letters*, **41**(4), 363-365.