An adaptive large neighbourhood search with MILP and heuristic repair operators for bus timetabling

Robin Gaborit[a,*], Evelien van der Hurk[a], Otto Anker Nielsen[a], Yu Jiang[a,b],
[a] Department of Technology, Management and Economics, Technical University of Denmark, Denmark
rpaga@dtu.dk, evdh@dtu.dk, oani@dtu.dk, yujiang@dtu.dk
[b] Lancaster University Management School, Lancaster University, United Kingdom
*Corresponding author
Extended abstract submitted for presentation at the 12th Triennial Symposium on Transportation Analysis conference (TRISTAN XII) June 22-27, 2025, Okinawa, Japan

Keywords: Metaheuristic, Bus timetabling, ALNS, Matheuristic, Operator selection

1 INTRODUCTION

Research goals: This study is motivated to leverage an efficient solution method for Lee *et al.* (2022)'s acyclic bus timetabling problem. The model considers passengers' entire journey experience composed of multiple travel cost components and time-dependent travel times and demand. Specifically, we resort to the Adaptive Large Neighbourhood Search (ALNS) method (Ropke and Pisinger, 2006) and consider the differences in computation times between operators.

Lee *et al.*'s model is very constrained, and defining performant heuristic operators is difficult. Thus, a major feature of our proposed ALNS is that it combines a pure heuristic operator and several operators repairing solutions with a Mixed Integer Linear Programming (MILP) solver. The latter make our ALNS a matheuristic.

The heuristic repair operator may not be as effective as the MILP repair operators in finding improving solutions, but it consumes much less computation time per iteration. The majority of studies on ALNS ignore computation time when setting the weights for operator selection. The few studies that acknowledge computation times fall into an intuitive pitfall (Adulyasak and Cordeau, 2014; Gullhav *et al.*, 2017; Laborie and Godard, 2007; Lei, Che and Van Woensel, 2024; Thomas and Schaus, 2018).

Contributions: The main contributions of this study include integrating MILP and heuristic repair operators for the ALNS, identifying a common pitfall when setting the weights of operators with different computation times, and devising a formula called the inverse-square rule to correct it.

2 SOLUTION METHOD

2.1 ALNS framework

Due to space limitations, the detail of the algorithm is not given in this abstract. Similar to standard ALNS, destroy-repair operator are assigned weights reflecting their performance. At each iteration, an operator is selected based on the weights and through a roulette-wheel principle to generate a candidate solution. Depending on the quality of the generated solution compared to the best and incumbent solutions, the candidate solution is accepted or not, and the weight of the corresponding operator is updated. Since our operators require very different computation times, we measure the time they consume and propose a formula to adjust the weights based on average computation time (section 2.3).

2.2 Destroy-repair operators

We design five destroy-repair operators. Four of them, namely *MILP_4runs*, *MILP_8runs*, *MILP_2routes8runs*, and *MILP_halfAllRuns*, are denoted as *MILP* repair operators. To repair the

solution, a restricted MILP problem is formulated by enforcing the non-destroyed decision variables to their current values in the complete formulation presented in Lee *et al.* (2022). This is done by adding equality constraints for each to fix their values. Then, a commercial solver is utilised to solve the restricted MILP problem with a time limit. The last operator, namely *Heur_1run1min*, is denoted as a heuristic repair operator. First, it repairs the departure time from the first stop by randomly adding or subtracting one minute from its value before destruction. Then, a dwell time is drawn for each subsequent stop. The MILP repair operators tend to produce higher-quality solutions than heuristic repair, but they are much more time-consuming.

2.3 Probability of selecting a destroy-repair operator

A widely adopted procedure is to use the following equation to compute the probability associated with each operator and then apply the roulette wheel selection to select an operator from a set of operators.

$$p_i = \frac{\omega_i}{\sum_{j\in\Omega}\omega_j} \ \forall i \in \Omega\,, \tag{1}$$

where Ω is the set of operators and ω_i is the weight associated with operator i.

To the best of our knowledge, only several ALNS algorithms have explored the idea of considering the computation time required by operators in operator selection (Adulyasak and Cordeau, 2014; Gullhav *et al.*, 2017; Laborie and Godard, 2007; Lei, Che and Van Woensel, 2024; Thomas and Schaus, 2018). Despite the variation in their formula, a common presumption embedded in these studies is that the weight is a function of the inverse of the computation time. However, we found that presuming the weight is a function of the inverse of the computation time may lead to undesirable behaviour of the solution method. Specifically, the computation time allocated in the algorithm may not be efficient because too much time would be spent on the slowest operators. The full paper will explain why setting weight as a function of the inverse of the computation time can lead to an undesirable distribution of computation time between operators and provide an illustrative example.

To address the above-mentioned issue, this study proposes the following formula called the inverse-square rule to set the weights of selecting an operator in the ALNS.

$$\overline{\omega_i} = \frac{\omega_i}{\Delta_i{}^2}, \ \forall i \in \Omega \tag{2}$$

where
$$\omega_i = (1-\eta)\omega_i + \eta\sigma, \forall i \in \Omega, \text{ and} \tag{3}$$

where ω_i and $\overline{\omega_i}$ are, respectively, the weights associated with operator i without and with considering computation time. To differentiate these two terms, $\overline{\omega_i}$ is called the revised weight in this paper. Δ_i is the average computation time associated with operator i, Ω the set of operators, σ the score, and η the reaction factor defined by $\min\left(\eta_{max} * \frac{1/|\Omega|}{p_i}, \eta_{max}\right)$, where p_i is the probability of selecting operator *i*. The full paper will give proof that Eq. (2) distributes computation time between operators proportionally to the ratio between weight and computation time, overcoming a pitfall in the existing literature.

3 NUMERICAL RESULTS

3.1 Case study

The present study focus on three bus lines of the Copenhaguen bus network. A total of nine instances were produced based on the Danish Rejsekort smart card transactions from 2014. The instances differ in the optimisation period, the number of passenger groups and their paths. S1, S2, and S3 denote the small instances, M1, M2, and M3 the medium instances, and L1, L2, and L3 the large instances.

3.2 Illustration of the proposed operator selection

Figure 1 illustrates the changes in the operator selection. Our solution method controls the distribution of running time between operators. At the beginning of the optimisation process, finding improving solutions is relatively easy, including for the heuristic repair operator. Consequently, since the heuristic repair operator is much faster than the MILP repair operator, most of the computation time is allocated to the MILP repair operator. As time passes, improving the incumbent solution becomes more difficult for the heuristic repair operators whereas the MILP repair operators are still able to frequently generate improving solutions. Thus, the share of time allocated to the MILP repair operators increases. After a while, almost all the computation time is allocated to the MILP repair operators.

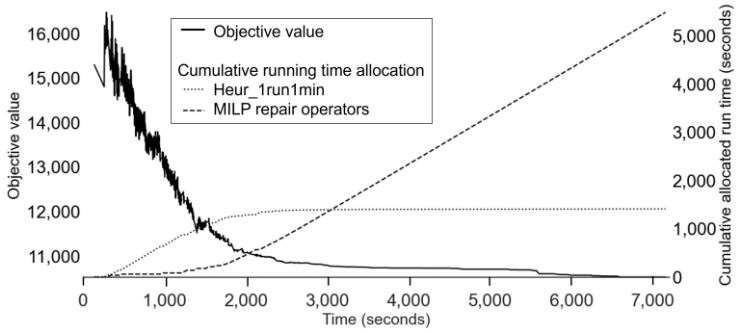

Figure 1 – *Illustration of the changes in operator selection*

3.3 Comparison to Gurobi

For each instance, we produce ten feasible solutions with our construction heuristic. They are used as ten different initial solutions. Then, we run Gurobi and our solution method ten times with a computation time of two hours, once for each initial solution.

After ten runs we obtain the average objective values using ALNS, $ALNS_{avg}$, and Gurobi, $Gurobi_{avg}$, Then, we calculate relative changes in the average objective values of ALNS compared to Gurobi expressed as $Gap = \frac{ALNS_{avg} - Gurobi_{avg}}{Gurobi_{avg}}$. A negative Gap value represents an improvement of ALNS compared with Gurobi. We also obtain the objective value using Gurobi without providing an initial feasible solution, denoted Gur^*, and the relative change between this value and $ALNS_{avg}$, expressed as $Gap^* = \frac{ALNS_{avg} - Gurobi^*}{Gurobi^*}$. The results are presented in Table 1.

Table 1 – *Results from Gurobi and our algorithm*

	S1	S2	S3	M1	M2	M3	L1	L2	L3
Gap	-10.7%	-4.3%	-4.7%	-34.2%	-37.9%	-32.6%	-33.8%	-36.4%	-33.8%
Gap^*	-2.9%	/	-4.7%	/	/	/	/	/	/

Remark: / indicates that Gurobi without an initial solution, is not able to generate a feasible solution after two hours

Our ALNS solution method outperforms Gurobi (with and without the initial solution) on every instance. The average objective value is at least 2.9% better than Gurobi on small instances and 32.6% better on medium and large instances.

3.4 Impact of the MILP repair and heuristic repair operators

We assess the effect of the MILP repair operators by removing them and comparing the results with the ones obtained with our base combination of operators (MILP repair and heuristic repair operators activated). We perform a similar experiment with the heuristic repair operator.

The solution method is run ten times for each combination and instance. The average objective value $Restricted_{avg}$ obtained after running each restricted combination is compared to the average

objective value given by the base combination $Base_{avg}$ through $Gap = \frac{Restricted_{avg} - Base_{avg}}{Base_{avg}}$. A positive value indicates that the solution method performs better when all operators are activated. The results are presented in Table 2.

Table 2 – *Comparison of the performance of different operator combinations*

Combination	S1	S2	S3	M1	M2	M3	L1	L2	L3
onlyMILPRepair	1.3%	1.0%	0.9%	**2.6%**	**2.2%**	**2.7%**	**4.8%**	**4.1%**	**4.6%**
onlyHeuristicRepair	**5.7%**	**5.6%**	**4.8%**	**4.7%**	**5.0%**	**4.3%**	**2.0%**	**2.1%**	1.5%

Remark: The percentage measures the changes in the objective value w.r.t. the base case (all five operators activated).). A positive value means that the performance is worse. The values greater than 2% are bolded.

Removing the heuristic repair operator is detrimental on every instance. The gap is at least 0.9% on every instance and 4.1% for the large ones. Removing all four MILP repair operators worsens the average results in every instance by at least 1.5%. The gap is at least 4.3% on the small and medium instances.

4 DISCUSSION

The results show that our ALNS outperforms Gurobi on every instance. Moreover, combining MILP repair operators and a heuristic repair operator is superior to including only one of the two types of operators. Ongoing experiments include evaluating the impact of each destroy-repair operator. We will also compare our mechanism setting the probability to select operators, namely the inverse-square rule, to other inverse-power formulas. In particular, we will provide evidence that setting the weights inversely proportional to the square of the computation time outperforms setting the weights inversely proportional to the computation time.

References

Adulyasak, Y. and Cordeau, J.F. (2014) 'Optimization-Based Adaptive Large Neighborhood Search for the Production Routing Problem'. doi: https://doi.org/10.1287/trsc.1120.0443.

Gullhav, A.N. *et al.* (2017) 'Adaptive large neighborhood search heuristics for multi-tier service deployment problems in clouds', *European Journal of Operational Research*, 259(3), pp. 829–846. doi: https://doi.org/10.1016/j.ejor.2016.11.003.

Laborie, P. and Godard, D. (2007) 'Self-Adapting Large Neighborhood Search: Application to Single-Mode Scheduling Problems'.

Lee, K. *et al.* (2022) 'Path-oriented synchronized transit scheduling using time-dependent data', *Transportation Research Part C: Emerging Technologies*, 136, p. 103505. doi: https://doi.org/10.1016/J.TRC.2021.103505.

Lei, J., Che, A. and Van Woensel, T. (2024) 'Collection-disassembly-delivery problem of disassembly centers in a reverse logistics network', *European Journal of Operational Research*, 313(2), pp. 478–493. doi: https://doi.org/10.1016/j.ejor.2023.07.008.

Ropke, S. and Pisinger, D. (2006) 'An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows', *Transportation Science*, 40(4), pp. 455–472. doi: https://doi.org/10.1287/trsc.1050.0135

Thomas, C. and Schaus, P. (2018) 'Revisiting the Self-adaptive Large Neighborhood Search', in W.-J. van Hoeve (ed.) *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 557–566. doi: https://doi.org/10.1007/978-3-319-93031-2_40.