Ikonen, Teemu; Harjunkoski, Iiro

Decision-making of online rescheduling procedures using neuroevolution of augmenting topologies

Published in:
Proceedings of the 29th European Symposium on Computer Aided Chemical Engineering

DOI:
10.1016/B978-0-12-818634-3.50197-1

Published: 01/01/2019

Document Version
Peer reviewed version

Please cite the original version:
Ikonen, T., & Harjunkoski, I. (2019). Decision-making of online rescheduling procedures using neuroevolution of augmenting topologies. In A. Kiss, E. Zondervan, R. Lakerveld, & L. Özkan (Eds.), *Proceedings of the 29th European Symposium on Computer Aided Chemical Engineering* (Vol. 46, pp. 1177-1182). (Computer-aided chemical engineering; Vol. 46). Elsevier. https://doi.org/10.1016/B978-0-12-818634-3.50197-1

Anton A. Kiss, Edwin Zondervan, Richard Lakerveld, Leyla Özkan (Eds.)
Proceedings of the 29th European Symposium on Computer Aided Process Engineering
June 16th to 19th, 2019, Eindhoven, The Netherlands.

Decision-making of online rescheduling procedures using neuroevolution of augmenting topologies

Teemu J. Ikonen[a] and Iiro Harjunkoski[a,b,*]

[a]*Aalto University, School of Chemical Engineering, Kemistintie 1, 02150 Espoo, Finland*
[b]*ABB Corporate Research, Wallstadter Str. 59, 68526 Ladenburg, Germany*
**iiro.harjunkoski@aalto.fi*

Abstract

Online scheduling requires appropriate timing of rescheduling procedures, as well as the determination of relevant horizon length. Optimal choices of these quantities are highly dependent on the uncertainty of the scheduling environment and may vary over time. We propose an approach where a neural network is trained to make online decisions on these quantities, as well as on the choice of the rescheduling method (mathematical programming or metaheuristics). In our approach, the neural network is trained using neuroevolution of augmenting topologies (NEAT) in a simulated environment. In this paper, we also optimize the rescheduling interval and horizon length of a conventional periodically occurring rescheduling on a dynamic routing problem. The resulting approach is the baseline for the development of the proposed neural network approach.

Keywords: online scheduling, horizon length, scheduling interval, neural network, NEAT

1. Introduction

Online scheduling of industrial processes is a demanding task, mainly due to a large number of solution candidates and uncertainty in scheduling parameters. Considering batch processes, the most common optimization approach in offline scheduling is mixed-integer programming (MIP), using modeling formalisms, e.g. resource-task network (Pantelides, 1994), state-task network (Kondili et al., 1993) or generalized disjunctive programming (Raman and Grossmann, 1994). Given a reasonably large amount of computational time, these approaches can often find the optimal solution. However, in online scheduling, the duration of the optimization process is constrained due to the moving time horizon, which significantly compromises the optimality of the solution. Another aspect to consider is that the later the optimization process is initiated, the more accurate are its scheduling parameters. This raises questions, such as what is the optimal rescheduling interval and what is a relevant length for the scheduling horizon. For a recent review of online scheduling, the reader may wish to consult the paper by Gupta et al. (2016).

We propose an approach where an artificial neural network trained to decide when to initiate a rescheduling process, what length to define for the time horizon and which optimization strategy (mathematical programming or metaheuristics) to use. In many previous studies, neural networks have been used to learn dispatching rules from the historical data of scheduling decisions (Priore et al., 2014). However, this approach can typically only mimic the historical decisions, and therefore the optimality of the scheduling solu-

tions is highly dependent on the quality of the given historical data. Our neural network approach operates at a higher level; it exploits the strength of MIP at finding the optimal, or near-optimal solution, and focuses on the allocation of the computational resources in the changing scheduling environment.

More specifically, we propose to use neuroevolution of augmenting topologies (NEAT), introduced by Stanley and Miikkulainen (2002), to train the neural network. NEAT is a genetic algorithm that simultaneously evolves the topology and weighting parameters of the neural network. Such et al. (2017) report the performance of the evolutionary neuroevolution algorithms to compare well against the gradient-based backpropagation algorithms. A key feature of NEAT is that the evolution process is initiated from very simple neural networks, the topology of which is then incrementally increased complexity during the evolution. The feature reduces unnecessary complexity of the final neural network, and is not possible using gradient-based algorithms. Recently, Hausknecht et al. (2014) applied NEAT to train a neural network to play 61 different Atari 2600 games, where the controls for the player are to move a joystick and press a button. The controls for our neural network are also of a similar low level of complexity (see the second paragraph), but the critical aspect is their timely execution.

In this paper, we first define a rescheduling problem (Section 2), suitable to be used as a test case for our neural network-based rescheduling approach. For the sake of simplicity, we define the underlying optimization problem to be a routing problem. However, our approach is also applicable to rescheduling of other industrial processes, e.g. batch processes. In Section 3, we tune the horizon length and rescheduling interval of the conventional periodically occurring rescheduling approach. This approach is a benchmark for our neural network-based approach. Finally, in Section, 4, we describe the intended input and output signals for the neural network.

2. Dynamic routing problem

Let us consider a square region, having dimensions 1000×1000 m, and a vehicle traveling in the region at a constant speed of $v = 10$ m/s. The purpose of the vehicle is to visit n sites in the region before site-specific due dates. Each site has an order date, at which the vehicle receives the location and due date of the site. The objective of the optimization problem is to minimize the delay sum of visiting the n sites. Since scheduling decision must be made with limited information, finding the optimal solution to the problem requires rescheduling.

As an optimizer, we here use ant colony optimization (ACO) (Dorigo and Gambardella, 1997), which is a probabilistic metaheuristic search method of finding good paths in a graph. In the method, ants communicate by laying pheromone on paths between nodes. The laid pheromone concentration of an ant is proportional to the objective function value of the route the ant traveled. When a new ant is at node i, it chooses the path to node j with probability

$$p_{ij} = \frac{\phi_{ij}^{\alpha} d_{ij}^{\beta}}{\sum_{i,j=1}^{n} \phi_{ij}^{\alpha} d_{ij}^{\beta}}, \tag{1}$$

where ϕ_{ij} and d_{ij} are the pheromone level and desirability[1] of the path (i, j). where α and β are influence parameters[2]. We use a population size of 200. We have implement ACO using the Python module ACOpy (Grant, 2018).

In this and the next section, we study periodic rescheduling with fixed interval and horizon length. At $t = 0$, the vehicle starts to follow an initial path, determined by a greedy algorithm (which is defined to always choose the node lying closest to the current node). The path is then updated when the first rescheduling procedure is finished. The computational budget of all the rescheduling procedures during the time span is restricted to 50 CPU seconds. The computational budget is distributed evenly between the rescheduling processes. Therefore, a frequent rescheduling will have a small computation budget per rescheduling process, and vice versa. Each rescheduling procedure is associated with times t_{info} and t_{exe}, the difference of which is the duration of the procedure. The rescheduling is conducted using the information available at t_{info}, whereas t_{exe} is the planned start time of the new schedule.

2.1. A simple numerical example

Let us next present a solution to a simple routing problem of nine sites. The sites are randomly distributed inside the region (Figure 1(a)), and their due dates are randomly selected from the time interval of [0, 500s] (Table 1). Further, the order dates of sites 1 to 7 are set to 0 s, while the order dates of sites 8 and 9 are randomly selected from the time interval of $[0, t_{\text{due}}]$, where t_{due} is the due date of the site (Table 1).

Table 1: Order and due dates for the sites in the simple numerical example.

site [-]	order date [s]	due date [s]	site [-]	order date [s]	due date [s]
1	0	46	6	0	400
2	0	198	7	0	346
3	0	343	8	4	43
4	0	14	9	185	439
5	0	279			

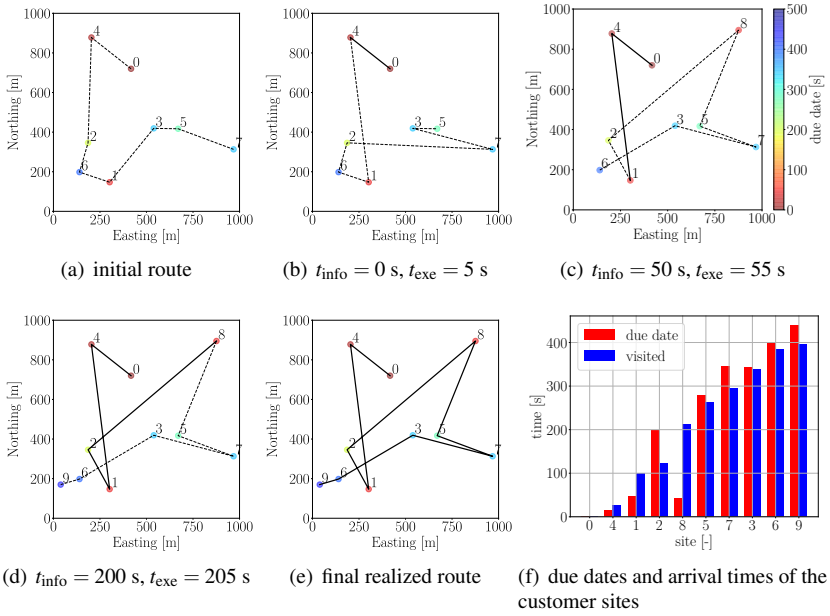
In this numerical example, we choose to use a scheduling interval of 50 s and horizon length of 500 s (i.e. the entire time span of the problem). Figure 1(a) shows the initial route (determined by a greedy search) for the vehicle. Further, Figures 1(b)-1(d) visualize intermediate states of the process. At $t_{\text{exe}} = 5$ s, the optimizer schedules the visit to the most critical site 1 before visiting sites 6 and 2. At $t_{\text{exe}} = 55$ s and $t_{\text{exe}} = 205$ s, the optimizer includes visits to sites 8 and 9, respectively, in the schedule. The locations and due dates of these sites were not known at the start of the time span. Figure 1(e) shows the final (realized) route of the vehicle. The delay sum of the final route is 235.30 s, which is caused by the vehicle failing to meet the due dates at sites 1, 4 and 8 (see Figure 1(f)).

3. Tuning of the rescheduling interval and horizon length

The purpose of the numerical example in the previous section was to provide an introduction to the dynamic routing problem, which we intend to study with the neural network

[1] In this work, we use the distance of a path as its desirability measure.

[2] In this work, we use values $\alpha = 1$ and $\beta = 3$.



(a) initial route (b) $t_{\text{info}} = 0$ s, $t_{\text{exe}} = 5$ s (c) $t_{\text{info}} = 50$ s, $t_{\text{exe}} = 55$ s

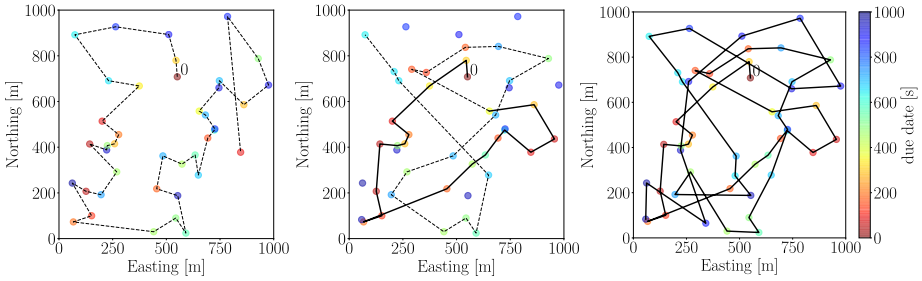
(d) $t_{\text{info}} = 200$ s, $t_{\text{exe}} = 205$ s (e) final realized route (f) due dates and arrival times of the customer sites

Figure 1: Representative states and the final results of the rescheduling procedure with nine sites to be visited. The locations and due dates of sites 8 and 9 are received at $t = 4$ s and 185 s, respectively, whereas the other orders are already known at $t = 0$. Continuous path represents realized route of the vehicle, and dashed line the scheduled route.

based approach. In this section, we examine a similar, but larger, routing problem with a total of 50 sites, and a time span of 1000 s. The locations and due dates of 40 sites are known at $t = 0$, whereas for the remaining 10 sites the information is obtained again at a random time point in the interval of $[0, t_{\text{due}}]$, where t_{due} is the due date of the site.

When using the periodically occurring rescheduling procedures, the optimal values for the rescheduling interval and horizon length are problem-dependent. Thus, in this section, we also use a grid search to systematically seek an optimized combination of the two parameters. The results of the grid search are shown in Figure 3. Out of the tested parameter combinations, the best final objective value (i.e. the delay sum) is obtained by the scheduling interval of 120 s and the horizon length of 500 s. We omit those parameter combinations, in which the scheduling interval is larger than the scheduling horizon (see the bottom right corner of Figure 3), as the vehicle would need to wait at certain sites for the schedule to be updated. Based on the results, the performance of the rescheduling procedure is poor for combinations lying close to this constraint, but also for combinations with very frequent rescheduling and long horizon length. In the former, the algorithm does not exploit enough information it has about the future, where as in the latter the algorithm does not have enough computational time to find good solutions (to relatively large scheduling problems).

Finally, let us examine the rescheduling procedure with the optimized rescheduling interval and horizon length. Figures 2 and 4 visualize the initial route, the intermediate state at



(a) initial route (b) $t_{\text{info}} = 240$ s, $t_{\text{exe}} = 245.56$ s (c) final realized route

Figure 2: Representative states and the final results of the rescheduling procedure with 50 sites.

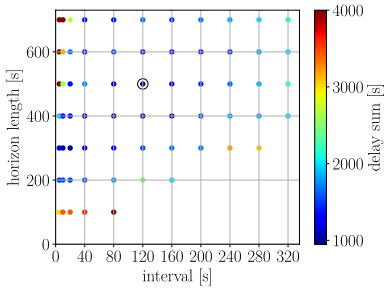


Figure 3: A grid search of best-performing rescheduling interval and horizon length. The best parameter combination is highlighted by a circle. Darkest red points exceed the scale.

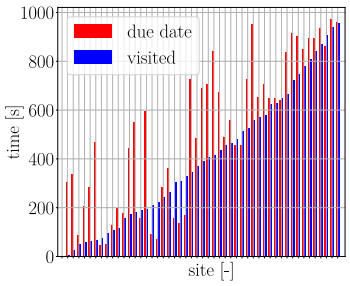


Figure 4: Due dates and arrival times of the final route (Figure 2(c)).

$t_{\text{exe}} = 245.56$ s and the final realized route of the rescheduling procedure. The delay sum of the final route is 992.89 s. Despite being a solution to a larger scale problem, the same features are also present here as in the simple example (Section 2.1): sites with urgent due dates are prioritized and, as the time proceeds, new orders are included in the schedule.

4. Proposed input and output signals

The periodically occurring rescheduling with tuned interval and horizon length seems to work reasonably well for these relatively small-scale scheduling problems. Another possible approach would be to trigger the rescheduling always when new information is obtained. However, the rescheduling decisions made by these approaches may be compromised, especially in larger problems, or if new information is obtained more frequently.

Thus, we propose the scheduling decisions to be made using a neural network that is trained by the NEAT algorithm. Figure 5 shows the input nodes we propose for the neural network. The input signals represent changes in the scheduling environment, the status of the ongoing rescheduling procedure, and the remaining computational resources.

At each evaluation, the neural network decides to either run a rescheduling procedure, using metaheuristics or mathematical programming, or to do nothing (depending which of the three top output nodes receives the highest signal). In addition, the neural network includes signals for the horizon length and allocated computational time for the possible rescheduling procedure.

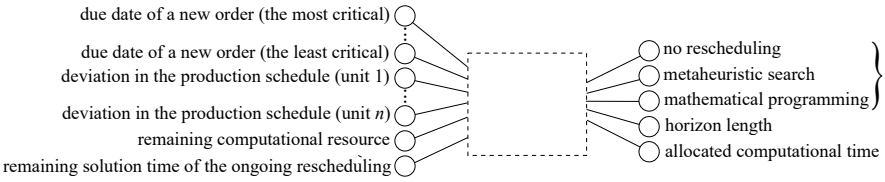


Figure 5: Proposed input and output signals for the neural network. Here, the neural network is depicted as a box as in NEAT its topology is decided during the training procedure.

5. Conclusions

We propose the rescheduling decision in online scheduling to be made by a neural network, trained using the NEAT algorithm. As a first step, this paper presents a dynamic routing problem suitable to be used as a demonstration and benchmarking problem for our approach. We optimized the rescheduling interval and horizon length of the conventional periodically occurring rescheduling procedure, in order to obtain a relevant baseline approach. The future work involves training and evaluation of the NEAT algorithm for the presented routing problem.

Acknowledgment: Financial support is gratefully acknowledged from the Academy of Finland project "SINGPRO", Decision No. 313466.

References

M. Dorigo, L. M. Gambardella, 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on evolutionary computation 1 (1), 53–66.

R. Grant, 2018. Acopy [accessed on the 15th of october, 2018]. `https://github.com/rhgrant10/acopy`.

D. Gupta, C. T. Maravelias, J. M. Wassick, 2016. From rescheduling to online scheduling. Chemical Engineering Research and Design 116, 83–97.

M. Hausknecht, J. Lehman, R. Miikkulainen, P. Stone, 2014. A neuroevolution approach to general atari game playing. IEEE Transactions on Computational Intelligence and AI in Games 6 (4), 355–366.

E. Kondili, C. C. Pantelides, R. W. H. Sargent, 1993. A general algorithm for short-term scheduling of batch operations–I. MILP formulation. Computers & Chemical Engineering 17 (2), 211–227.

C. C. Pantelides, 1994. Unified frameworks for optimal process planning and scheduling. In: Proceedings on the second conference on foundations of computer aided operations. Cache Publications New York, pp. 253–274.

P. Priore, A. Gómez, R. Pino, R. Rosillo, 2014. Dynamic scheduling of manufacturing systems using machine learning: An updated review. AI EDAM 28 (1), 83–97.

R. Raman, I. E. Grossmann, 1994. Modelling and computational techniques for logic based integer programming. Computers & Chemical Engineering 18 (7), 563–578.

K. O. Stanley, R. Miikkulainen, 2002. Evolving neural networks through augmenting topologies. Evolutionary computation 10 (2), 99–127.

F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, J. Clune, 2017. Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567.