Applicability of Optimizing Control for Dynamic Discrete-Event Systems

LENKO GRIGOROV grigorov@cs.queensu.ca
School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada
KAREN RUDIE rudie@ee.queensu.ca
Department of Electrical and Computer Engineering, Queen's University, Kingston, Ontario K7L 3N6, Canada

***Abstract*— The results of a simulation of a new near-optimal online control method for dynamic discrete-event systems is presented. The performance of the suggested control algorithm is compared to the performance of a previous online control algorithm which does not use optimality information. A case-specific *value function*, needed for the control optimization, is described. The paper provides insight into the applicability of the new approach for the control of dynamic discrete-event systems.**

I. INTRODUCTION

A wide variety of systems, such as factory robots and communications protocols, can be suitably described as discrete-event systems (DESs). Possible approaches to their control are standard supervisory control [5] and online control with limited lookahead [2]. Of the two, only the latter is applicable to dynamic discrete-event systems (DDESs)—systems which can vary with time [4]. Such systems are constituted of separate, small DES modules, which are combined together using synchronous product [1]. The systems vary in that, after each occurrence of an event, they may be comprised of different modules.

In the aforementioned theories, control is concerned mainly with the acceptability (legality) of the resulting system behavior. In [4], a modification to the online control algorithm is proposed so that control becomes not only acceptable, but also near-optimal with respect to user-selected criteria. The user of a system defines a *value function*, $v : L(G) \rightarrow \mathbf{R}$, such that it returns the value of each event string that can be generated. This function is used to evaluate nodes in the look-ahead tree built by the online controller. By convention, v should return $-\infty$ for illegal strings.

In this paper we present the results of a proof-of-concept simulation which was carried out to compare the performance of the suggested control method [4] to the performance of the original online control method [3]. It provides insight into the applicability of the new approach for the control of DDESs.

II. SIMULATION SETUP

For the purpose of comparing the two control methods, the example provided in [3] is used. This example is chosen for two reasons: the system setup is such that it immediately fits in the DDES paradigm, and it offers a very convenient way to compare the new algorithm to the original algorithm.

The dynamic system under consideration consists of a number of trains, each modeled as a separate module. Each train can travel along tracks, which are connected as shown in Fig. 1. There is a tunnel in section 2 of each of track 1 and 4. Trains can enter the system from stations 1 and 2 and can leave the system from stations 2 and 3. The number of trains is not known a priori and trains can enter and leave the system at different times (see [3]), which makes the system a DDES.

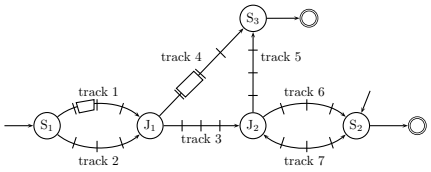

Fig. 1. Train system. Stations are denoted by S_i.

The value function v used to optimize the system control is based on the legality specifications in [3] and it returns $-\infty$ when an event string is illegal. In addition, it is used to attempt to fulfill the following specifications:

1) Trains should move through the system aiming to reach stations. Thus, trains which arrive at a station different from their starting station contribute the value 200 to the overall value of the string, while arrivals at the starting station contribute negatively with a value of -200.
2) The use of the tracks and tunnels should be balanced. Thus, for every use of a section of a track, the value of $1 + \ln m$ is subtracted, where m stands for the number of times the particular section has been used. Furthermore, for every string, the difference between the tunnel usage times 10 is subtracted.
3) Trains should move in an interweaved fashion (i.e., should not wait for too long before they move). Thus, for each train it is calculated how many time intervals have passed since it has moved last and then all these intervals are summed up and subtracted.

The simulation consists of two parts. First, the original online control algorithm is run to collect performance information not collected in [3] but important for the comparison of the two methods. Then, the modified algorithm is run to collect the same performance information:

- Average number of trains during the simulation.
- The value of the generated strings at the end of each simulation.
- The number of trains which arrive at a station during each simulation.

TABLE I

SIMULATION RESULTS

Length of run	Tree depth	Trains (avg)	Value of generated string (min)	Value of generated string (avg)	Value of generated string (max)	Arrivals (avg)	Decision time per step (ms) (min)	Decision time per step (ms) (avg)	Decision time per step (ms) (max)	Inspected nodes per step (avg)	Reused nodes per step (avg)
(a) Using the original online control algorithm [3]											
40	10	4.3	-1095.8	-583.0	90.5	1	0	132.8	1612	107.1	1.2
40	20	4.4	-1668.2	-725.3	135.4	0.8	0	500.7	7711	423.9	1.3
100	10	4.9	-2218.4	-1546.9	-750.8	2.4	0	581.9	39147	316.6	1.4
(b) Using the DDES optimal control algorithm [4]											
40	2	5.2	-216.5	-39.0	287.6	0.7	0	44.4	170	25.2	0.7
40	3	6.5	-70.9	79.2	296.9	1.4	10	158.1	911	193.3	4.3
40	5	6.4	-69.1	124.9	217.8	1.7	30	2502.9	34810	3487.6	102.1
40	7	5.2	210.9	227.0	243.1	2	60	11039.1	104721	14893.0	307.5
100	5	8.4	927.7	1183.3	1291.1	7.8	20	14593.7	85823	13992.5	155.3

- The decision time for every control step.
- The number of tree nodes inspected at every control step.
- The number of tree nodes whose value was reused (i.e., had been computed previously) for every control step. Please note that this is different from the re-utilization statistic provided in [3], where the average ratio of the portion of the look-ahead window that survives between steps is shown.

For each pair of tree depth and simulation length, ten simulation runs are executed and the results are averaged for these runs. The only exceptions are the minimal and maximal values which reflect the overall minimal or maximal value for the given set of runs.

Both algorithms are implemented in the Java programming language. Other implementations may run faster or slower; thus, the time spent by the algorithm at each control step should be used for comparative purposes within this simulation only. In eight out of the ten runs with a length of forty and a tree depth of seven, the Java Virtual Machine ran out of memory (64 MB) before the completion of the simulation. Information from only the two successful runs is included in the table with results.

III. RESULTS

The information gathered for the original online control method is shown in Table I-a. The results are comparable to the results presented in [3] with the exception of the decision time per step which improved with the new implementation. At the end of each run, the string produced is evaluated with the value function to serve as a basis for comparison with the optimizing control method. The average value of the strings is negative. This is due to the randomness in the execution of events—as long as an event is legal, it is considered for execution and there is no attempt to guide the system along a path that is valuable for the user. Naturally, the increase of tree depth does not have a beneficial effect on the performance of the algorithm.

The information gathered for the newly proposed control method is shown in Table I-b. Overall, the optimizing control algorithm outperforms the original online control algorithm. The execution of events is targeted and thus the resulting value of the event strings is much greater. Furthermore, as expected, the greater the depth of the look-ahead tree, the better is the performance of the algorithm—both in terms of the string value and the average number of arriving trains. On the other hand, the optimal control algorithm requires more computational resources than the original algorithm, both in terms of time and space. Almost every branch of the look-ahead tree has to be examined fully and thus the decision time and the number of nodes created are considerably greater than in the original control algorithm. However, the values are still good enough for the practical use of the system in this example. Furthermore, as seen, the proposed control algorithm performs well even with very shallow look-ahead trees; this all depends on the specific value function used.

IV. CONCLUSION

In this paper we presented the result from a simulation comparing the performance of online control of dynamic discrete-event systems with and without optimality information. It was shown that optimizing control increases significantly the value of strings generated by the system. However, this comes at the cost of greater time and space complexity. The computational demands of this method may render it inapplicable to very large systems. On the other hand, the original online control method may still perform well with larger systems. This leads us to believe that ultimately a combination of the two approaches would offer the best performance.

REFERENCES

[1] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Norwell, Massachusetts, USA: Kluwer Academic Publishers, 1999.

[2] S.-L. Chung, S. Lafortune, and F. Lin, "Limited lookahead policies in supervisory control of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1921–1935, 1992.

[3] ——, "Supervisory control using variable lookahead policies," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 4, pp. 237–268, 1994.

[4] L. G. Grigorov, "Control of dynamic discrete-event systems," Master's thesis, Queen's University, Kingston, Ontario, Canada, 2004.

[5] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal of Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.