ISSUES IN OPTIMAL CONTROL OF DYNAMIC DISCRETE-EVENT SYSTEMS

Lenko Grigorov [*] **Karen Rudie** [**]

[*] grigorov@cs.queensu.ca
School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada
[**] rudie@ee.queensu.ca
Department of Electrical and Computer Engineering, Queen's University, Kingston, Ontario K7L 3N6, Canada

Abstract: We define the notion of Dynamic Discrete-Event Systems, a class of time-varying systems, and present a simple approach to optimal control of such systems. More specifically, we use limited-lookahead online control and an algorithm which tries to maximize the benefit of the executed sequences of events, while at the same time ensuring that unwanted (illegal) sequences are avoided. We use examples to illustrate the different types of problems that can arise if such control is used, for example, overspecialization and failure to take advantage of available resources. These issues are used to formulate a list of desirable properties for a new algorithm that optimizes the control of dynamic systems. *Copyright ©2005 IFAC*

Keywords: Discrete-event systems, Time-varying systems, Dynamic behaviour, On-line control, Optimal control, Control algorithms, Uncertain systems

1. INTRODUCTION

One of the accepted ways to model real-world systems is using a set of states and transitions between them. This is used mainly when a system, such as a factory robot or a communication protocol, can be suitably described as a discrete-event system (DES). One possible approach to the control of such systems is the Standard Supervisory Control of DES paradigm (Ramadge and Wonham, 1989). Despite the numerous advantages of this theory, its applications to real-world problems is very limited due to the daunting computational requirements. Another approach to the control of DES is found in (Chung *et al.*, 1992; Chung *et al.*, 1994; Kumar *et al.*, 1998), describing the limited lookahead control policy, or online supervisory control. At the expense of worse reliability, it brings the advantages of reduced computational costs and applicability to a broader range of real-world systems, including systems which can vary with time. While many modifications of online control are proposed, little attention has been devoted to the study of online control of time-varying discrete-event systems.

In the aforementioned theories, control is concerned mainly with the acceptability of the resulting system behavior (or the "legality" of the executed sequences of transitions between states). Other works propose modifications to the control algorithms so that control becomes not only acceptable, but also optimal with respect to some selected criteria (Sengupta and Lafortune, 1998; Kumar and Garg, 1995). Unfortunately, these approaches are designed only for standard control, using stable (static) systems. Cost is associated with transitions between states and the algorithms avoids sequences of transitions which will result in high accumulated costs. In (Chen and Lin, 2001), a method which can be used to construct an optimal online controller which maximizes the performance of the system while minimizing the associated cost is presented. Even though this and similar methods are immediately applicable to time-varying DESs, there are some issues which arise from their time-varying nature and which are not considered in these works.

When time-varying systems are considered, using marked languages rarely makes sense: due to the limited lookahead window and due to the dynamics of the system, the controller may frequently get into a position when the system cannot be guided to execute a marked string. However, it is not desirable to immediately terminate control with an error message: in the next time period the system may alter and the controller may be able to lead the system to a desired state. Instead of using marked languages, a desirability measure on strings can be used to indicate what behavior of the system is most beneficial.

The limit which is chosen for the lookahead calculations of a control action might have crucial impact on the optimality of the control. When static systems are taken into account, the greater the depth of the lookahead tree, the better are the predictions of the controller and the control is more optimal. However, when time-varying systems are considered, a deep lookahead window may lead to overspecialization of the prediction and the resulting control becomes unrealistic.

Closely related to the above is the problem of greediness vs. long-term planning in control optimization. For static systems, long-term planning is better. However, for dynamic systems the greedy approach may produce better results, since it utilizes the available resources promptly before the system has the opportunity to evolve to a point where a resource is no longer available.

In this paper we define a class of time-varying discrete-event systems and we present a simple online algorithm for optimal control of such systems. Then, we give examples which illustrate the problems that may arise if this algorithm is used. Finally, we present specifications that should be met by an algorithm which optimizes the control of dynamic discrete-event systems.

2. PRELIMINARIES

We assume that the reader is acquainted with (Ramadge and Wonham, 1989) and related work, where discrete-event systems are modeled as finite-state automata. Standard notation is used in the paper: G is the finite-state automaton for the system, Σ is the set of events, Σ_c and Σ_{uc} are the subsets of controllable and uncontrollable events, respectively, $L(G)$ is the language generated by the system, $\overline{L}$ is the prefix-closure of the language L, $K \subseteq L(G)$ is the legal language, and the elements of Σ^* are usually denoted by the lowercase letters s, t, u...

Supervision of DES consists of ensuring that only legal sequences occur in the system, i.e., that whatever s is executed, $s \in K$. The online controller (Chung *et al.*, 1994) can be defined as a function which returns the subset of events which should be enabled (all other events will be disabled) after a given string so that the system behavior is restricted to strings in K. Note that Σ_{uc} will always be contained in this subset. The limited-lookahead controller operates by constructing a tree of all the possible continuations of the executed prefix, up to a certain depth. Then it recursively explores the branches of this tree to determine which nodes correspond to illegal sequences. The algorithm propagates the information back to the root and determines which events need to be disabled to prevent the occurrence of illegal strings. However, because the tree depth is limited, the controller might not have the information needed to make the correct decisions. This problem is discussed in (Chung *et al.*, 1992; Chung *et al.*, 1994) and related work.

There are many ways that one can model time-varying systems. Here we present a discrete-time module-based approach which can be used with a wide range of systems and which, at the same time, is suitable for the purposes of control. We call systems modeled using this approach dynamic discrete-event systems (DDESs). Time is considered to be discrete—it increases by one after each occurrence of an event. At each time period the system may consist of the composition of a different set of modules which are combined using synchronous product (see (Cassandras and Lafortune, 1999)). Thus, the system varies in time in that old modules may disappear and new ones may appear.

Formally, a DDES can be modeled as follows. Let $M_i = \{M_{1i}, M_{2i}, \ldots, M_{ni}\}$ be some set of DES modules and let $\|M_i$ denote the composition of all elements of M_i. Then DDES $G = \{(\|M_i, i) \mid i \in \{0, 1, \ldots\}\}$ and the system at time i is $G_i = \|M_i$.

3. SIMPLE OPTIMAL CONTROL ALGORITHM

In this work we will discuss issues which arise when optimization of the control of DDES is considered. We understand optimization as the attempt by the controller to guide the execution of events so that the resulting strings have minimal cost or maximal benefit to the user or owner of the system. Thus we can define a value function $v : L(G) \to \mathbf{R}$ which will return the value of a particular string (and the function, of course, can be defined to return the cost of strings instead). This function can be as complex as necessary, however, for our purposes we will consider a very simple form:

$$\begin{aligned} v(s) &= -\infty && \text{if } s \notin \overline{K}, \\ v(s) &= \sum_{s=\sigma_1\sigma_2\ldots\sigma_n} c(\sigma_i) && \text{otherwise,} \end{aligned} \tag{1}$$

where $c(\sigma_i)$ is the single-event value of elements of Σ. The single-event values can be negative for costs and positive for payoffs. The online controller will explore the branches of the lookahead tree and compute the value function for each possible continuation. The values will be propagated back to the root using a simple approach: if all events leading from a node are controllable, the maximal value from the children is taken (since the controller will be able to disable the unwanted events). Otherwise, if there are uncontrollable events leading from the node, the minimal of the values of the children via the uncontrollable events will be taken (since the controller will not be able to prevent the most costly behavior through uncontrollable events). As well, the controller will not explore branches further on if they have infinite costs. The algorithm, shown in Fig. 1, is a slight modification of the algorithm proposed in (Chung *et al.*, 1994).

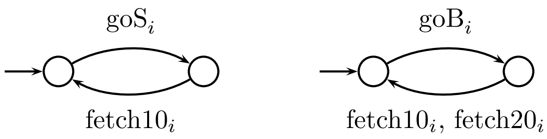
```
benefit-to-go($x$,$h$) {
/* $h$ is the string generated so far */
$\Sigma_{out}$ = events going out of $x$
if($\Sigma_{out} \cap \Sigma_{uc} = \emptyset$) {
   forall($\sigma \in \Sigma_{out}$) {
      $y$ = stateVia($\sigma$)
      if($h\sigma$ is illegal)
         $v_\sigma = -\infty$
      elseif($y$ hits boundary)
         $v_\sigma = v(h\sigma)$
      else
         $v_\sigma$ = benefit-to-go($y, h\sigma$) }
   return $\max_{\sigma \in \Sigma_{out}}(v_\sigma)$ }
else {
   forall($\sigma \in (\Sigma_{out} \cap \Sigma_{uc})$) {
      $y$ = stateVia($\sigma$)
      if($h\sigma$ is illegal)
         $v_\sigma = -\infty$
      elseif($y$ hits boundary)
         $v_\sigma = v(h\sigma)$
      else
         $v_\sigma$ = benefit-to-go($y, h\sigma$) }
   return $\min_{\sigma \in (\Sigma_{out} \cap \Sigma_{uc})}(v_\sigma)$ }
}
```

Fig. 1. Simple optimal control algorithm

This algorithm is very simple and it would work for stable systems, since it retrieves as much optimality information as possible from the limited lookahead tree. Unfortunately, some issues arise when it is used with DDESs. We proceed by giving examples of these issues.

4. EXAMPLE SYSTEM DESCRIPTION

Before we present the examples, we will provide a brief description of the system which we will explore. Let us consider that we have a company which needs supplies, for example wooden logs. The company uses the logistic services of a provider and the contract is such that we can rent one truck at a time. The provider has two types of trucks available—a small truck (see Fig. 2(a)) and a big truck (see Fig. 2(b))—which can bring ten logs or either ten or twenty logs, respectively.



(a) small truck ST_i (b) big truck BT_i

Fig. 2. DES models of trucks

The system consists of the synchronous product of these modules. The number and type of trucks available at any given time is not known in advance. Thus the system is inherently dynamic. Let us consider that our company needs a resupply of exactly forty logs and we would like to have a controller which will guide the system so that we achieve our goal.

The legality specification for the system is given by the following two rules:

(1) After one truck goes to fetch supplies, we cannot send another truck (Fig. 3).

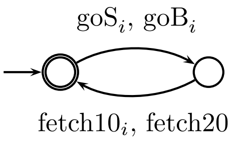


Fig. 3. First restriction of the legal language, where $i \in \{1, 2, \ldots\}$.

(2) The number of logs fetched is forty: $\forall s \in K$,

$$\#_{\text{fetch10}}(s) \times 10 \; + \; \#_{\text{fetch20}}(s) \times 20 = 40,$$

where $\#_\sigma(s)$ denotes the number of occurrences of σ in s.

The values of single events are as follows:

- $c(\text{goS}_i) = -100$
(we pay \$100 to rent the small truck)
- $c(\text{goB}_i) = -150$
(we pay \$150 to rent the big truck)
- $c(\text{fetch10}_i) = 500$
(the revenue we get from every log is \$50)
- $c(\text{fetch20}_i) = 1000$

and the value function v will be computed as described previously by (1).

All events in the system are controllable. Even though the proposed algorithm allows for uncontrollable events, the exclusive use of controllable events renders the illustrative examples clearer.

5. EXAMPLES

Example 1

In this example we will illustrate the effect of an overly limited lookahead capability. For this purpose, let us limit the prediction of the controller to just one step ahead. At the start the system will have a small truck and a big truck available.

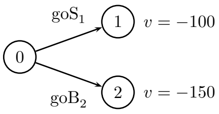


Fig. 4. *Time 0* (one small truck, one big truck)

Since the prediction capability is very limited, the tree is very shallow and simple. The cost of renting a big truck is greater than the cost of renting a small truck. Thus, at *time 0* (Fig. 4), the controller, whose task is to optimize the behavior of the system, chooses to disable the event goB$_2$ and leave only the less costly goS$_1$.

Since our company needs forty logs, and since sending a small truck four times to bring the logs is more expensive than sending a big truck twice for the same number of logs, one would correctly observe that it is preferable to send the big truck. This judgment is based on the knowledge of what one expects to happen after a type of truck is dispatched. Unfortunately, the controller is much more limited—it can only foresee one step ahead. The fundamental problem illustrated here is that an online controller cannot provide optimal control if it cannot observe far enough along event sequences to compute both the relevant costs and the relevant payoffs.

Example 2

In this example we will illustrate the effect of overly lenient lookahead capability. For this purpose, the controller will be able to predict four steps ahead. The system will be similar to the one in the previous example. At the beginning there will be a small and a big truck available. At *time 2* (after one round-trip) there will be only a small truck available. At *time 4* (after two round-trips) there will be only a big truck available.

In this example the trees are much deeper and more complex because the prediction capability is stronger (Fig. 5). At *time 0* (Fig. 5(a)), the controllerrecognizes that sending the big truck is preferable, since it is cheaper to bring twenty logs at a time (the value function v examines sufficiently long portions of the event sequences). Thus goS$_1$ is disabled and the big truck is dispatched.

At *time 2* (Fig. 5(c)), after one round-trip, only the small truck is available (for example, someone else might have rented the big truck). After the small truck fetches ten logs, the only available truck is the big truck (for example, the small truck might need maintenance). Thus at *time 4* (Fig. 5(e)), the controller has to enable goB$_2$ even though the truck will be used to fetch only ten logs.

After bringing the last ten logs, the goal is accomplished. As one can observe, however, the system ended up incurring a greater cost than necessary. Had we used the small truck at *time 0*, the payoff of fetching forty logs would have been

$$v(\mathrm{goS_1 fetch10_1 goS_1 fetch10_1 goB_2 fetch20_2}) = 1650,$$

while the payoff in this example is

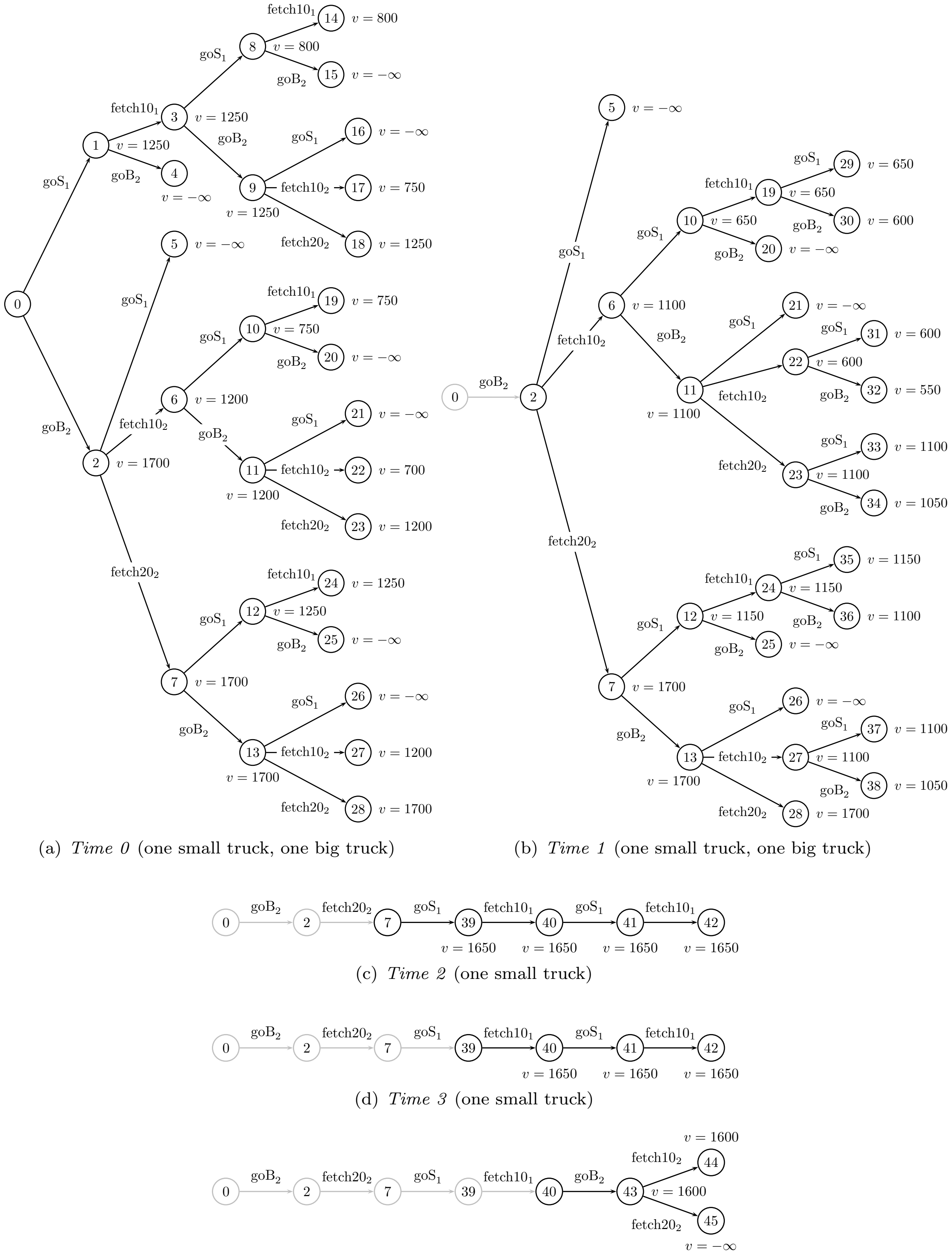
$$v(\mathrm{goB_2 fetch20_2 goS_1 fetch10_1 goB_2 fetch10_2}) = 1600.$$

In other words, the solution produced by the controller is not optimal. Unlike the situation in example 1, the problem is not caused by the lack of information: the controller has sufficient lookahead capability. This time the cause is the availability of incorrect information. Since the system is dynamic, basing control decisions on predictions of a too distant future will most likely result in incorrect assumptions. In this example, at *time 0* the system assumes the big truck will be available at *time 2* and the computations of string values are based on this. The fundamental problem illustrated here is that the lack of a good model of the changes of a system renders far-reaching predictions inherently unreliable.

Example 3

In this example we will illustrate why an optimization approach with long-term planning may not be suitable for the control of dynamic systems. Let us assume that the system will have a scheduler which can postpone the sending of a truck using a wait event, where $c(\mathrm{wait}) = 0$. There is a small truck available at *time 0* and it will not be available at *time 1* if it is not used.

In this setting, at *time 0* (Fig. 6) the strings "goS$_1$fetch10$_1$wait" and "wait goS$_1$fetch10$_1$" have the same value and thus the controller will choose randomly between them. As a result, once out of two tries the system will miss the chance to utilize the available equipment. The fundamental problem illustrated here is that, due to the unreliability of dynamic systems, a greedy approach where the systems attempts to execute short valuable strings



(a) *Time 0* (one small truck, one big truck)

(b) *Time 1* (one small truck, one big truck)

(c) *Time 2* (one small truck)

(d) *Time 3* (one small truck)

(e) *Time 4* (one big truck)

Fig. 5. Lookahead trees for Example 2

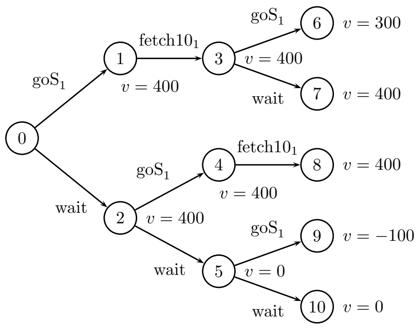


Fig. 6. *Time 0* (one small truck)

may be preferable over well-planned strategies that reach far in the future.

6. SPECIFICATIONS FOR AN OPTIMIZING CONTROLLER OF DDES

When designing an algorithm for the optimizing online control of dynamic discrete-event systems, the following specifications should be met:

- Legality constraints should be observed and the system should avoid executing illegal strings;
- From all legal strings, the controller should choose to execute the string that would have the most value to the user of the system;
- The controller should be conservative with risk-taking (i.e., worst-case should be assumed when uncontrollable strings are evaluated);
- The controller should be optimistic when there is no risk (i.e., the best case should be pursued when the strings are controllable);
- Exploration of subtrees of the lookahead tree should be limited only to nodes which may influence the control decision.

The above are standard requirements also for controllers of static systems. However, as shown in the examples from Section 5, there is a need to introduce new specifications for the optimizing control of dynamic systems:

- The controller should account for the variability of dynamic systems and should put less emphasis on strings too far in the future;
- The controller should attempt to utilize the available resources: it should prefer strings that benefit the user sooner rather than later.

7. CONCLUSION

In this paper we defined the notion of Dynamic Discrete-Event Systems and presented a simple algorithm for optimal online control. We used examples to illustrate the issues which may arise when this algorithm is applied to the control of time-varying systems. Since the system can change as time progresses, a very deep lookahead tree may become a disadvantage due to the overspecialization of the control decision. Furthermore, long-term planning may be a worse choice than assuming a greedy attitude. We conclude the discussion with a list of properties that an algorithm optimizing the control of dynamic systems should have.

Our current research includes the development of an algorithm which satisfies the specifications listed in Section 6. The algorithm takes into account the unreliability of DDESs and it uses a normalization on the string values, depending on how deep the lookahead tree is explored. Furthermore, it uses estimation of future node values to minimize the number of nodes that need to be explored.

8. ACKNOWLEDGEMENTS

This work was supported, in part, by the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

Cassandras, C. G. and S. Lafortune (1999). *Introduction to Discrete Event Systems*. Kluwer Academic Publishers. Norwell, Massachusetts, USA.

Chen, Y.-L. and F. Lin (2001). An optimal effective controller for discrete event systems. In: *Proceedings of the 40th IEEE Conference on Decision and Control*. Vol. 5. pp. 4092–4097.

Chung, S.-L., S. Lafortune and F. Lin (1992). Limited lookahead policies in supervisory control of discrete event systems. *IEEE Transactions on Automatic Control* **37**(12), 1921–1935.

Chung, S.-L., S. Lafortune and F. Lin (1994). Supervisory control using variable lookahead policies. *Discrete Event Dynamic Systems: Theory and Applications* **4**, 237–268.

Kumar, R. and V. K. Garg (1995). Optimal supervisory control of discrete event dynamical systems. *SIAM Journal of Control and Optimization* **33**, 419–439.

Kumar, R., H. M. Cheung and S. I. Marcus (1998). Extension based limited lookahed supervision of discrete event systems. *Automatica* **34**(11), 1327–1344.

Ramadge, P. J. and W. M. Wonham (1989). The control of discrete event systems. In: *Proceedings of the IEEE*. Vol. 77. pp. 81–98.

Sengupta, R. and S. Lafortune (1998). An optimal control theory for discrete event systems. *SIAM Journal of Control and Optimization* **36**(2), 488–541.