A LEARNING AUTOMATA APPROACH FOR INPUT-RATE CONTROL IN
COMPOSABLE CONVEYOR SYSTEMS

A Thesis

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Chandana Cheeralu

May, 2011
A LEARNING AUTOMATA APPROACH FOR INPUT-RATE CONTROL IN
COMPOSABLE CONVEYOR SYSTEMS

Chandana Cheeralal

Thesis

Approved:       Accepted:

Advisor
Dr. Shivakumar Sastry

Department Chair
Dr. Alex De Abreu Garcia

Committee Member
Dr. Hamid Bahrami

Dean of the College
Dr. George K. Haritos

Committee Member
Dr. Arjuna Madanayake

Dean of the Graduate School
Dr. George R. Newkome

Date
Simple and effective control techniques are necessary to improve the Quality of Service (QoS) in a variety of composable and reconfigurable networked systems. Traditional approaches to improve QoS are model-based and rely on representations of the state of the system, at multiple levels. This thesis presents a Learning Automata based approach to improve QoS that does not need to explicitly represent or estimate system state.

A Learning Automaton (LA) is a model for adaptive decision-making that uses only stochastic feedback, also called reinforcement, from the environment and does not rely on detailed models or estimation of parameters. It learns to choose the optimal actions from a finite set of actions using such interactions with its environment.

This investigation focused on a class of composable conveyor systems that represent a confluence of embedded and real-time systems, wireless communication, sensing and actuating devices, and networking technologies. In such systems, input-rate, i.e., the rate at which parts are injected into the system affects QoS. The input-rate controller presented in this thesis learns the “correct” rate of injection at an input based on noisy feedback on the latency and throughput of parts that moved through
the system recently. When it is important to deliver each part at its output before its deadline is elapsed, an admission controller must be used to determine whether the part can be so delivered or not. Preliminary results on value of a probabilistic admission controller are also presented in this thesis. Simulation results confirm the performance of these controllers. These results motivate several investigations in the future to design better admission controllers, route controllers and congestion controllers for composable and networked systems that are based on simple adaptive mechanisms.
ACKNOWLEDGMENTS

First and the foremost, I would like to express my sincere gratitude to my advisor Dr. Shivakumar Sastry for giving me an opportunity to work on the most interesting and challenging areas in the field of the networked embedded automation. With his enthusiasm, inspiration, and great effort to explain things clearly helped me quickly understand the underlying objectives for this research.

Besides my advisor, I would like to thank my committee members: Dr. Hamid Bahrami and Dr. Arjuna Madanayake for their suggestions and insightful comments on this thesis. I would like to thank my colleagues in the Complex Engineered Systems Laboratory (CESL): Kranthi Mamidisetty, Branden Archer, Mukesh Chippa, Chidi Anozie and Sanchita Subedi for their help and support. A big thank you to Kranthi and Branden for helping me with critical issues during the simulator design.

Loving thanks to all my friends for their constant support and encouragement. Lastly, and most importantly, I wish to thank my parents for their warm wishes, and above all of us God, for their blessings. To them I dedicate this thesis.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>viii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
</tbody>
</table>

## CHAPTER

I. INTRODUCTION ............................................. 1  
   1.1 Contributions ........................................ 5  
   1.2 Overview ........................................... 5  

II. BACKGROUND ............................................. 6  
   2.1 Coupled Conveyors .................................... 6  
   2.2 Learning Algorithm ................................... 7  
   2.3 Related Work ......................................... 9  

III. DESIGN AND IMPLEMENTATION OF DISCRETE-EVENT SIMULATOR ............................................. 11  
   3.1 Simulator Design ...................................... 11  
   3.2 Simulation Approach .................................. 13  
   3.3 QoS Metrics .......................................... 13  

vi
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>14</td>
</tr>
</tbody>
</table>

Temporal trace of Parts that arrived from $I_1$. 

viii
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Conveyor Systems are composed using instances of Segments and Turns. Parts move from Inputs to Outputs.</td>
<td>7</td>
</tr>
<tr>
<td>3.1</td>
<td>Discrete Event Simulation Framework</td>
<td>12</td>
</tr>
<tr>
<td>4.1</td>
<td>A representative conveyor system in which parts move over fixed paths.</td>
<td>16</td>
</tr>
<tr>
<td>4.2</td>
<td>System performance for parts from $I_3$ when parts are injected at Poisson(0.1) (top panel) and Poisson(0.4) (bottom panel) at $I_1, I_2, I_4$ and $I_5$.</td>
<td>18</td>
</tr>
<tr>
<td>4.3</td>
<td>Evolution of action probabilities for $L_{RP}$ controller.</td>
<td>21</td>
</tr>
<tr>
<td>4.4</td>
<td>Performance of LA input-rate controller when the rates at other inputs are Poisson(0.1) (top panel) and Poisson(0.4) (bottom panel) with $\lambda_1 = 0.003$ and $\lambda_2 = 0.0003$.</td>
<td>22</td>
</tr>
<tr>
<td>4.5</td>
<td>Performance of LA input-rate controller when the rates at other inputs are Poisson(0.1) (top panel) and Poisson(0.4) (bottom panel) with $\lambda_1 = 0.03$ and $\lambda_2 = 0.003$.</td>
<td>23</td>
</tr>
<tr>
<td>4.6</td>
<td>Performance of LA input-rate controller when the rates at other inputs are Poisson(0.1) (top panel) and Poisson(0.4) (bottom panel) with $\lambda_1 = 0.01$ and $\lambda_2 = 0.001$.</td>
<td>24</td>
</tr>
<tr>
<td>4.7</td>
<td>When the topology is disrupted and repaired, $L_{RP}$ based input-rate controller appropriately tracks the changes.</td>
<td>26</td>
</tr>
<tr>
<td>4.8</td>
<td>Evolution of action probabilities for $L_{RP}$ controller.</td>
<td>27</td>
</tr>
<tr>
<td>4.9</td>
<td>$L_{R-P}$ based input-rate controller effectively selects actions.</td>
<td>28</td>
</tr>
</tbody>
</table>
5.1 The ratio $C = \frac{L_{act}}{\theta_{min}}$ settles around an equilibrium for different release rates at the inputs.

5.2 There is a steep increase in the number of parts that did not meet their deadline around the converged value of $C$ shown in Figure 5.1.

5.3 The probabilistic admission controller admits more parts than the deterministic admission controller that meet their deadline.
CHAPTER I
INTRODUCTION

Simple and effective control techniques are necessary to improve the Quality of Service (QoS) in a variety of composable and reconfigurable networked systems. Traditional approaches to improve QoS are model-based and rely on representations of the state of the system, at multiple levels. For example, buffer capacity and size at the nodes represent state information in communication networks. Even in static systems, it is often difficult to estimate or maintain adequate state information. When the structure of the system changes dynamically, either because of variations in demand or because of operational issues such as maintenance and failures, it is difficult to construct and maintain models of the system that can support effective decision-making. This thesis presents a simple adaptive method to improve QoS that does not need to explicitly represent or estimate system state. The approach is based on Learning Automata.

A Learning Automaton (LA) is a model for adaptive decision-making that uses only stochastic feedback, also called reinforcement, from the environment and does not rely on detailed models or estimates of parameters [1]. It learns to choose the optimal actions from a given, finite, set of actions called its action set, by relying only on noisy feedback from its environment. At each time instant, the LA
randomly chooses an action from its action set based on its current action probability distribution. Using the feedback from the environment, the LA updates the action probability distribution and uses the updated distribution to select the next action. The algorithm that is used to update the action probabilities is called the Learning Algorithm. The learning algorithm used in this investigation is presented in Chapter 2.

The investigation was focused on a class of composable conveyor systems that represent a confluence of embedded and real-time systems, wireless communication, sensing and actuating devices, and networking technologies [2]. Such systems are important for moving material and parts in automation applications and must deliver predictable QoS under dynamically changing situations.

The QoS was characterized in terms of end-to-end latency and throughput. *End-to-end latency* is the time required to move a part from its input to its output. *Throughput* is the number of parts that arrive at an output in a given duration of time. Congestion is an important factor that degrades QoS; it is caused either because of the arrival rates of the parts, poor design of the system topology, or because of failures in one or more units. The relatively large latency of parts, concurrent part streams, and overlaps across different part streams complicate the feedback that is necessary to monitor and regulate the movement of parts in the system.

When the topology of the conveyor systems is non-trivial, latency and throughput achieved in the systems are interrelated in complicated manners. For example, consider a simple conveyor system in which there is only one path from a single input
to a single output. In such a system, the simplest approach would be to inject one part at the input and wait for this part to arrive at the output before injecting the next part at the input. Since there is only one part in the conveyor system, there is no chance for congestion to occur. Hence, every part will achieve the lowest end-to-end latency. Clearly, such a scenario is not realistic because the throughput achieved in the system will be very low. To improve throughput, the injection strategy could be modified to inject new parts as soon as the first part has moved out of the way of the next part. This strategy will achieve the highest throughput that is possible, assuming there is an adequate supply of parts at the input. Suppose now that there are multiple inputs and multiple outputs in the system. If there are independent non overlapping paths between every pair of input and output in the system, then the modified injection strategy discussed above can be used. However, such a topology presents a significant risk because if one unit along a path is disrupted because of a failure, the entire path is disrupted. Further, if the number of parts that are available at the inputs are such that the conveyor system units are not fully utilized, then the cost of such a topology increases. For these reasons, it is necessary to use conveyor system topologies in which multiple paths share common units. In such systems, the topology of the system impacts the manner in which latency and throughput are interrelated. When the system topology gets more complicated, it is not always feasible to analytically characterize the complicated relationship between latency and throughput.
To find an acceptable trade-off between latency and throughput, as in other networked systems, the problems of admission control, input-rate control, route control etc. must be addressed to mitigate the effects of congestion. Among these problems, the focus of this investigation was on input-rate control. This problem is challenging because the units are regulated autonomously in a decentralized manner. It is difficult to build realistic models of the system and its operating environment. Also, constraints on network bandwidth and power preclude global information sharing. Thus, one of the central issues in the design of learning schemes is to use limited information about the rest of the system while making the control decisions.

This thesis presents the design of an input-rate controller that is based on Learning Automata. This controller learns the “correct” rate of injection at an input based on noisy feedback on the latency and throughput of parts that moved through the conveyor system recently. Simulation results demonstrate the effectiveness of the approach. Results that demonstrate the behavior of multiple controllers at different inputs of the conveyor system are also presented. In addition, this thesis presents preliminary results that demonstrate the need for an effective admission controller in Chapter 5.
1.1 Contributions

The specific contributions of this thesis are:

1. Design and implementation of a discrete-event simulator for composable conveyor systems.

2. Tools for extracting QoS metrics from simulation trace files.

3. Design and validation of input-rate controller that improves QoS.

This is the first known use of learning automata to improve QoS in conveyor systems. The approach is well suited for conveyor systems in which the topology changes dynamically and, hence, is likely to stimulate further investigations. The simulator, QoS extraction tools, and the insights from designing the input-rate controller provide a foundation for the design of new learning automata based controllers for admission control, route control, and congestion control in the future.

1.2 Overview

After presenting the background in Chapter 2, the design and implementation of the discrete-event simulator is presented in Chapter 3. The specific problem being addressed is also described in this chapter. The design and evaluation of the input-rate controller is presented in Chapter 4. Preliminary results on admission controller are presented in Chapter 5. Finally, Chapter 6, presents the conclusions.
CHAPTER II
BACKGROUND

This chapter presents background on composable conveyor systems, the classical learning algorithms used in this investigation, and related work in the area.

2.1 Coupled Conveyors

The conveyor systems considered here are composed using two kinds of units called *Segments* and *Turns*. These systems move parts from inputs (\(I\)) to outputs (\(O\)). Each unit has a fixed number of sensors, actuators, and predefined behaviors [3, 2]. A Segment moves a part from its upstream end to its downstream end. Input and Output units are Segments that can move parts only in one direction. A Turn has four ports that can be configured to either receive parts or send parts. A Turn can handle only one part at a time; hence, when two or more parts simultaneously arrive at a Turn, only one part is accepted by the Turn. Congestion occurs at the upstream units that precede the Turn units.

A conveyor system is a specific composition of instances of these units. Figure 2.1 shows an example system that has two inputs and two outputs. Two paths merge at \(T_1\) and the path is split at \(T_2\). These conveyor systems have been used for several previous investigations in reliability and availability analysis of conveyor
systems [4, 5, 6], performance of distributed real-time systems [7, 2], diagnosing congestion [8], and model-driven performance analysis [9].

Figure 2.1: Conveyor Systems are composed using instances of Segments and Turns. Parts move from Inputs to Outputs.

The structure of these conveyor systems can be modeled as a directed graph $G = (U, E)$. The nodes of $G$, $u_i \in U$ represent the units; an edge $(u_i, u_j) \in E$ represents that a part can move from $u_i$ to $u_j$. Parts that arrive via $I_k \in I$ are delivered to $O_j \in O$ along a path $P(I_k, O_j) = < u_1 = I_k, u_2, \cdots , u_n = O_j >$, where $u_i \in U$. Such paths can either be pre-computed when the system cannot be reconfigured, or discovered and maintained when the system is reconfigurable using standard shortest path algorithms. For a path, $P_i$, $\theta_{\text{min}}(P_i)$ represents the minimum end-to-end latency along $P_i$.

2.2 Learning Algorithm

The following is a simple linear algorithm for updating the action probability distribution. Let $\{\alpha_1, \alpha_2, \cdots , \alpha_n\}$ be the finite set of actions and let $p_i(k)$ represent the
action probability corresponding to \( \alpha_i \) at step \( k \), \( i = 1, 2, \ldots, n \). Let \( \alpha(k) \) denote the action selected by LA and let \( \beta(k) \in [0, 1] \) represent the feedback from the environment at step \( k \). Let \( \alpha(k) = \alpha_i \). Then, the action probabilities are updated as follows:

\[
\begin{align*}
    p_i(k+1) &= p_i(k) + \lambda_1 \beta(k)(1 - p_i(k)) - \lambda_2(1 - \beta(k))p_i(k) \\
    p_j(k+1) &= p_j(k) - \lambda_1 \beta(k)p_j(k) + \lambda_2(1 - \beta(k))(\frac{1}{n-1} - p_j(k))
\end{align*}
\]  

(2.1)

where, \( \lambda_1 \) and \( \lambda_2 \) are parameters of the algorithm that affect the ‘step-size’ in learning.

The reinforcement, \( \beta(k) \), is a stochastic signal that tells the automaton how good is the action \( \alpha(k) \). The objective for the automaton is to find the action for which the expected value of reinforcement would be maximum. Since there is no model of the environment, the probability distributions that govern the generation of the reinforcement signal are unknown. Hence the LA uses the learning algorithm to evolve the action probability distribution to maximize the expected reinforcement.

To intuitively understand the above learning algorithm, suppose the feedback is binary, i.e., \( \beta(k) \in \{0, 1\} \); a higher value of \( \beta(k) \) is desirable. Then, whenever an action choice results in ‘good’ feedback (\( \beta(k) = 1 \)), the probability of the selected action is increased and that of the other actions are decreased; if the feedback is ‘bad’ (\( \beta(k) = 0 \)), the probability of the selected action is decreased and that of all other actions is increased. The algorithm ensures that \( \sum_i p_i(k) = 1, \forall k \). This algorithm is called the \textit{Linear Reward-Penalty (LRP)} algorithm. If \( \lambda_2 = 0 \) in this
algorithm then it is known as the Linear Reward-Inaction ($L_{R-I}$) algorithm [1]. If the step-size parameters are sufficiently small then, under certain assumptions on the stationarity of the environment, these learning algorithms converge to an action probability distribution that maximizes expected value of reinforcement [1].

2.3 Related Work

The problem of input-rate control is addressed in the literature as a mechanism for mitigating congestion in networked systems [10, 11, 12, 13]. The adaptive source rate control scheme in [10] considers channel condition, the transport buffer occupancy and the delay constraints and aims to efficiently utilize the channel. The rate control scheme reported in [11] uses dummy packets to probe the network condition. This approach is suitable for real-time interactive applications in networks with high bandwidth-delay products and high bit error rates. An approach to alleviate incident-induced traffic congestion utilizes models for time-varying relationships of traffic and a control algorithm that estimates control variables [12]. An input-rate control scheme that relies on a model for channel dynamics is reported in [13].

Several techniques have been reported in the literature to address admission control in a variety of systems [14]. In wireless mesh networks, admission controllers regulate the transmission of new messages based on the current load and complete, or partial, information about the network traffic. A technique that relies on cliques of graphs is reported in [15]; sources calculate an admission ration and use this to admit new messages. The use of Bayesian learning to detect a new user in a CDMA
system is reported in [16]; decisions are made based on the past information about
the system and inferring the state of the system. Other admission controllers, such
as ones based on game theory [17], capacity analysis [18], and stochastic models [19]
are also reported in the literature.

An interesting approach suitable for multi-stage systems is reported in [20].
They infer the future utilization of the system if an item is admitted to the system.
The resources required to handle every item that arrives must be determined before
the item is admitted. Only items that do not violate a utilization criteria that is
specific to the anticipated path along which the item is expected to traverse. Since
the admission controller needs complete information about the system, this approach
is best suited when a single admission controller can make decisions for all inputs of
the conveyor system.

Learning automata have been used for adaptive routing in networks [21] and
for adaptive decision making in communication networks (e.g., [22]). Other appli-
cations of Learning automata include [23, 24, 25]. However, there are no reported
applications of learning automata for the kind of networked embedded systems that
considered in this thesis.
CHAPTER III
DESIGN AND IMPLEMENTATION OF DISCRETE-EVENT SIMULATOR

A discrete event simulator was designed for the conveyor systems using the OM-Net++ framework [26]. This chapter describes the design of the simulator, the simulation approach and the method by which QoS metrics were extracted from the simulation runs.

3.1 Simulator Design

This simulator enhanced the design reported in [27]. In contrast to the three units that were considered in the original design, the conveyor systems that are now considered only required two kinds of units — namely, Segments and Turns. Consequently, the design of the turn units were more sophisticated and required the redesign of the simulator.

Consistent with the design in [27], the current simulator also relies on the event-queue provided by the OMNet++ framework as illustrated in Figure 3.1. The OMNet++ framework provided the event queue and a collection classes that were useful for carrying out the simulations. One of the supported class, called cSimpleModule, was extended to design new classes for Segments and Turns. Input and Output units
were implemented as a specialization of Segments. The topology description files were also created for each conveyor system topology that was tested in the investigation.

The simulator was changed to produce a temporal trace of when parts enter and leave each unit in the system. New source modules were developed to inject parts at the inputs at different rates to ensure Poisson arrivals. The LA based controllers that are described in Chapter 4 and Chapter 5 were all developed in this investigation.
3.2 Simulation Approach

Simulations were carried out for a fixed number of slots. Each slot represented about five minutes of actual system time. In each slot, parts were injected at the inputs in two ways. First, parts were injected under a Poisson process with rate $\mu$ parts per second. This method of injection is represented as $\text{Poisson}(\mu)$ in this thesis. The second method is to inject parts at any specific constant rate.

In the simulator, the end-to-end latency and throughput for a batch of parts from a given input $I_k$ in a particular slot were monitored. By first injecting parts at all the inputs at the highest rate, the maximum latency of the parts, $L_{\text{max}}$, from $I_3$ to $O_2$ was determined. For each part in the simulation, $L_{\text{act}}$, was computed as the difference in the time between when the part arrived at $O_2$ and the time at which it entered $I_3$. The average of the ratio $L_{\text{act}}/L_{\text{max}}$, over all the parts in a slot, is the normalized latency reported in the results. The average throughput was computed by counting the parts that arrived at the sink over a unit time. The throughput reported is only for the parts that arrived from a particular input and is not the throughput at the sink from all inputs. Aggregate values of latency and throughput were used to generate the reinforcement feedback for the LA controllers as described in Chapter 4.

3.3 QoS Metrics

Table 3.1 shows an example temporal trace for parts in the simple conveyor system shown in Figure 2.1. All the parts that are shown in this table arrived via input $I_1$. 

13
These parts traverse along the units $T_1$, $S_2$, $T_2$, and $S_3$ before arriving at $O_1$. The rows in the table correspond to the units. For each unit, the time at which the part arrives on the unit is shown in the row marked “in”. The time at which the part exits that unit is shown as “out”. The columns correspond to different parts. Parts were injected into $I_1$ at Poisson(0.4).

Table 3.1: Temporal trace of Parts that arrived from $I_1$.

<table>
<thead>
<tr>
<th>Module name</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
<th>$P_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1 – in$</td>
<td>0.003</td>
<td>1.992</td>
<td>5.131</td>
<td>7.439</td>
<td>9.407</td>
</tr>
<tr>
<td>$S_1 – out$</td>
<td>2.017</td>
<td>4.339</td>
<td>7.145</td>
<td>10.073</td>
<td>12.021</td>
</tr>
<tr>
<td>$T_1 – in$</td>
<td>2.017</td>
<td>4.339</td>
<td>7.147</td>
<td>10.073</td>
<td>12.021</td>
</tr>
<tr>
<td>$T_1 – out$</td>
<td>3.178</td>
<td>5.5</td>
<td>8.306</td>
<td>11.234</td>
<td>13.182</td>
</tr>
<tr>
<td>$S_2 – in$</td>
<td>3.178</td>
<td>5.5</td>
<td>8.306</td>
<td>11.234</td>
<td>13.182</td>
</tr>
<tr>
<td>$S_2 – out$</td>
<td>5.193</td>
<td>7.517</td>
<td>10.298</td>
<td>13.244</td>
<td>15.193</td>
</tr>
<tr>
<td>$T_2 – in$</td>
<td>5.193</td>
<td>7.517</td>
<td>10.298</td>
<td>13.244</td>
<td>15.193</td>
</tr>
</tbody>
</table>

Note that the end-to-end latency, i.e., $L_{act}$ can be computed as the difference between when a part arrives at $O_1$ and the time at which it was injected at $I_1$. The temporal trace enables one to compute unit-level, path-level, or system-level utilization for the conveyor systems.
CHAPTER IV

INPUT-RATE CONTROL

The objective for input-rate controller is to accept parts into the system based on the current status of traffic in the system. Because the system topology is reconfigurable and the actual arrival of parts from other inputs is not known in advance, an on-line technique that can dynamically adjust the input-rate based on some noisy feedback regarding the current traffic is necessary. This chapter presents an LA based input-rate controller.

The approach pursued in this thesis was empirical, rather than theoretical, and was based on the discrete-event simulator described in the previous chapter. All the explorations were carried out in the context of the representative conveyor system topology that is described in the next section.

4.1 The Representative Conveyor System

To understand the nature of randomness in the conveyor systems considered and to test the viability of the LA based methods, the representative conveyor system shown in Figure 4.1 was used. Such a system is useful for sorting packages and has five inputs and three outputs.
The paths over which parts from each input traveled were fixed. Parts from $I_1$ and $I_2$ were sent to $O_3$; parts from $I_1$ were sent via the turns $T_1, T_2, T_5, T_7$ and $T_{11}$ via the segment $S_{29}$ and parts from $I_2$ were sent via the turns $T_1, T_2, T_4, T_8, T_9$ and $T_{11}$. Parts from $I_4$ were sent to $O_2$ via the turns $T_6, T_7, T_8$ and $T_9$. Parts from $I_5$ were sent to $O_1$ via the turns $T_6, T_7, T_3$ and $T_4$. Parts from $I_3$ were sent to $O_2$ via the turns $T_5, T_6, T_7, T_8$ and $T_9$. This conveyor system captures most of the challenges presented by this class of systems.

Figure 4.1: A representative conveyor system in which parts move over fixed paths.
4.2 Baseline Performance

A baseline performance for the system was established before validating the LA based controllers. The baseline performance is shown in Figure 4.1. This allowed us to know the best rate of injection at $I_3$ when arrivals at all other inputs are Poisson. Keeping Poisson($\mu$) at all other inputs and a constant rate at $I_3$, the throughput and the normalized end-to-end latency for parts going from $I_3$ to $O_2$ was observed in the simulator described in Chapter 3. Figure 4.2 illustrates the QoS achieved when the constant injection rate at $I_3$ was varied. With a low injection rate at the other inputs, i.e., Poisson(0.1), the throughput of parts from $I_3$ increases steadily and then levels off as the release rate at $I_3$ increases. The best throughput and latency were achieved when the release rate is about 0.6 pps. When the injection rate at the other inputs was high, i.e., Poisson(0.4), latency for parts from $I_3$ increased sharply even for a relatively low release rate at $I_3$. The throughput also leveled off at a lower value of 0.2 pps. The best compromise between latency and throughput was when the rate at $I_3$ is about 0.2 pps.
Figure 4.2: System performance for parts from \( I_3 \) when parts are injected at Poisson(0.1) (top panel) and Poisson(0.4) (bottom panel) at \( I_1, I_2, I_4 \) and \( I_5 \).
4.3 LA for input-rate control

An LA based input-rate controller was designed and implemented at the input $I_3$ for the system shown in Figure 4.1.

4.3.1 Action Set

Since an LA selects one out of a finite set of actions, a discretized set of rates, \{0.1, 0.2, \ldots, 0.8\}, was used as the actions at the input. Initially the action probabilities for all actions were equal (i.e., $p_i(0) = 0.125, \forall i$). In each slot, the controller stochastically selected a rate (an action) and injected parts at this rate. It received reinforcement feedback and updated its action probabilities. The action, i.e., injection rate, for the next slot was based on these updated action probabilities.

4.3.2 Reinforcement Feedback

Feedback is a critical factor that guides the learning algorithm. Feedback is complicated in these conveyor systems because of two factors. First, there is a delay between the time when an action is performed at an input and the time when feedback corresponding to such an action can be received at the input; this delay is because of the physical movement of the parts through the system. After an action is selected by the controller, several parts move through the system before any feedback can be available. Second, because the paths from multiple inputs merge, and because the parts in the system may have been admitted as a result of different actions, the feedback available is co-mingled.
The reinforcement feedback, \( \beta(k) \), was computed as follows. Recall that the objective is to minimize the deviance from the minimum end-to-end latency while maintaining a high throughput. Let \( \beta_1 = \frac{\theta_{\text{min}}}{L_{\text{act}}} \) and \( \beta_2 \) represent the ratio of the observed throughput to the maximum possible throughput. In slot \( k \)

\[
\beta(k) = 1 - \frac{(1 - \beta_1)^2 + (1 - \beta_2)^2}{2}
\]  

(4.1)

The LA used \( \beta(k) \) to update the action probabilities using either the \( L_{R-P} \) or the \( L_{R-I} \) algorithms that were discussed in Chapter 2.

4.4 Evaluation

Several simulations were carried out to evaluate the performance of the input-rate controllers. The controller was located at \( I_3 \) and parts were injected into the other inputs at Poisson(\( \mu \)). Each simulation was carried out for 10,000 slots.

4.5 Performance of \( L_{R-P} \) based Controller

An \( L_{R-P} \) based input-rate controller was implemented with \( \lambda_1 = 0.003 \) and \( \lambda_2 = 0.0003 \). The evolution of the action probabilities for this \( L_{RP} \) controller are shown in Figure 4.3. Note that the probability for one of the actions is markedly different than the others. This indicates that the LA based controller converges.

The performance achieved when parts were injected into the other inputs at Poisson(0.1) is shown in the top panel of Figure 4.4. The throughput and latency
achieved by the controller were close to the best possible as can be seen from the baseline performance shown in the top panel of Figure 4.2. The LA based controller achieved a normalized latency of about 0.25 and a normalized throughput of about 0.5 both of which were close to the best possible. More importantly, at the end of 10,000 slots it was observed that the action with highest probability (with a value of about 0.5) of the LA controller corresponds to release rate of 0.6 which was the best release rate at $I_3$ in this case as can be seen from Figure 4.2. The bottom panel of Figure 4.4 shows the performance of the controller when parts were injected at Poisson(0.4) at the other inputs. Once again, comparing with the baseline performance shown in Figure 4.2, it is seen that the LA controller achieves good throughput and latency. In
this case, the actions with highest action probabilities were corresponding to release rates 0.1 and 0.2; these probabilities were 0.33 and 0.3, respectively. Note that the best rate for this case is between 0.1 and 0.2 as can be seen from the baseline performance (cf. Figure 4.2, bottom panel).

Figure 4.4: Performance of LA input-rate controller when the rates at other inputs are Poisson(0.1) (top panel) and Poisson(0.4) (bottom panel) with $\lambda_1 = 0.003$ and $\lambda_2 = 0.0003$. 
The effect of the learning parameters \( \lambda_1 \) and \( \lambda_2 \) were also evaluated through simulation. Figure 4.5 shows the latency and throughput achieved when \( \lambda_1 = 0.03 \) and \( \lambda_2 = 0.003 \). These values are an order of magnitude larger than the values used to obtain the results shown in Figure 4.4.

Figure 4.5: Performance of LA input-rate controller when the rates at other inputs are Poisson(0.1) (top panel) and Poisson(0.4) (bottom panel) with \( \lambda_1 = 0.03 \) and \( \lambda_2 = 0.003 \).
Figure 4.5 shows the latency and throughput achieved when $\lambda_1 = 0.01$ and $\lambda_2 = 0.001$. Comparing the results in Figure 4.4, Figure 4.5 and Figure 4.6, it can be concluded that smaller values of the learning parameters yield less variability in the system performance in successive slots.

Figure 4.6: Performance of LA input-rate controller when the rates at other inputs are Poisson(0.1) (top panel) and Poisson(0.4) (bottom panel) with $\lambda_1 = 0.01$ and $\lambda_2 = 0.001$. 
4.6 Tracking Topology changes

Figure 4.7 illustrates how the $L_{R-P}$ based input-rate controller responds to changes in the system topology. In this simulation, units $S_4$ and $S_9$ shown in Figure 4.1 were forced to fail at slot 3000. In their failed state, these units did not accept any parts from their upstream turn, $T_2$, thus, eliminating parts from inputs $I_1$ and $I_2$ from the system. At slot 6000, these units were allowed to recover by clearing the fault. As shown in Figure 4.7, the input-rate controller at $I_3$ appropriately tracked the changes to the topology.

Before the failure of the units, in the LA controller, the action with highest probability was rate of 0.1 (and its probability was 0.22). After the failure, the action probability corresponding to release rate of 0.4 steadily increased to 0.35 and became the highest among all action probabilities. After the units recovered in slot 6000, the action probability corresponding to release rate 0.1 once again became the highest value at 0.2. Thus the controller was able to increase throughput from $I_3$ when parts from $I_1$ and $I_2$ were removed from the system. This adaptation came about simply due to LA learning algorithm without any additional information given to the controller. It may also be noted from the figure that the controller also attempted to minimize the latency appropriately.

Figure 4.8 shows the action probabilities associated with each of the actions. Note that at around slot 3000, the highest probability is associated with the action that corresponds to a release rate of 0.1. At slot 3000, this value starts to decrease
Figure 4.7: When the topology is disrupted and repaired, $L_{R-P}$ based input-rate controller appropriately tracks the changes.

and shortly thereafter, the probability associated with the action that corresponds to a release rate of 0.4 at $I_3$ is the highest. Around slot 6000, these values revert back to values that are similar before the disruption occurred. This confirms that the tracking demonstrated in Figure 4.7 is indeed because of learning.
Figure 4.8: Evolution of action probabilities for $L_{RP}$ controller.
4.7 Performance of $L_{R-I}$ based Controller

Figure 4.9 illustrates the performance of the input-rate controller that was based on the $L_{R-I}$ algorithm to update the action probabilities. Parts were injected at the other inputs at Poisson(0.4). The controller evolved to a state where it selected the action corresponding to release rate 0.1 with probability 0.44 and release rate 0.2 with probability 0.39. These actions correspond to the best choice when the injection rates are Poisson(0.4) as seen from the bottom panel of Figure 4.2. The $L_{R-I}$ based controller achieved a slightly better latency compared to the controller based on the $L_{R-P}$ algorithm.

Figure 4.9: $L_{R-I}$ based input-rate controller effectively selects actions.
4.8 Input-rate Controllers at Multiple Inputs

In the final experiment three LA controllers, each based on the $L_{R-I}$ algorithm, were used at the inputs $I_1$, $I_3$ and $I_5$. The purpose of this simulation was to understand whether such a decentralized control can learn good combination of input rates at multiple inputs. Figure 4.10 shows the performance achieved by these three controllers. As can be seen from the throughput and latency achieved, two controllers appeared to coordinate well and converged to selecting a single action. The third controller (at $I_5$), however, did not converge. Consequently, the latency for the parts from this input varied significantly. It was also observed that in each of these cases, the probability of one of the actions is significantly more than all others thus indicating that multiple controllers are converging to an action combination.
Figure 4.10: Performance of LRI Controllers at multiple inputs.
4.9 Discussion

This thesis did not address the theoretical issues of convergence or optimality of these adaptive strategies. When the controller was used at only one of the inputs and the when the statistics of traffic at other inputs can be assumed to be stationary, the effective stochastic environment as seen by the LA controller would most likely be stationary; in this case, the results in [1] can be utilized to establish the convergence of the controller to optimal rates.

When multiple controllers are used at different inputs, then, effectively, there is a game situation where multiple agents are acting in a decentralized fashion, each trying to optimize its own utility. Further analysis would be needed to understand the kind of equilibrium strategies available in such a game and whether the learning algorithms are capable of learning suitable equilibrium strategies. Such theoretical analysis is also needed to establish guarantees on the expected performance that can be achieved with such on-line controllers.

The results in this chapter collectively demonstrate that LA based input-rate controllers are effective online mechanisms for achieving good QoS in composable and reconfigurable conveyor systems.
CHAPTER V

SIMPLE ADMISSION CONTROL

The input-rate controllers attempted to minimize the latency and maximize throughput for all the parts in the system. In contrast, when every part has a unique deadline before which it must be delivered to its output, an admission controller is necessary to determine whether or not to admit a particular part to the system. This chapter presents a simple probabilistic admission controller that highlights a peculiar aspect of these conveyor systems that warrants further investigation in the future.

Suppose the conveyor system always had only a single part moving, the end-to-end latency is the minimum time, $\theta_{\text{min}}$, required to move along the path. This is because this part would never have to contend for resources with any part. Naturally, such an approach will result in very low throughput and utilization. For this reason, it is important to design an admission controller that can make suitable admission decisions while maintaining reasonable throughput. To make such decisions, the admission controller must learn the state of the traffic in the system. Recall that since each unit of the conveyor system is being regulated autonomously, it is challenging to make such a decision with limited information.
5.1 Natural Equilibrium

To understand the nature of the random environment in the conveyor system empirically, simulations were carried out using the system topology shown in Figure 4.1. Parts were injected at Poisson(0.4) at the inputs $I_1, I_2, I_4$ and $I_5$. By injecting parts at $I_3$ with a constant rate of 0.4 pps, the ratio

$$C = \frac{L_{act}}{\theta_{min}}$$

was computed for each part from $I_3$ to $O_2$.

Figure 5.1 shows the average value of $C$ computed over a time window. Notice that the value of $C$ changed a little around a fixed value that depending on the rate at which parts were injected into the system at the other inputs. As illustrated,

![Figure 5.1: The ratio $C = \frac{L_{act}}{\theta_{min}}$ settles around an equilibrium for different release rates at the inputs.](image)

Figure 5.1: The ratio $C = \frac{L_{act}}{\theta_{min}}$ settles around an equilibrium for different release rates at the inputs.
it appeared that there is a “natural” equilibrium for the conveyor system, and this equilibrium suggests a simple admission controller.

5.2 Performance of Simple Admission Controllers

Suppose the deadline of a part is $k \cdot \theta_{\text{min}}$ for some constant $k$. Two simple admission controllers were implemented. The first was a deterministic controller that admitted a part to the system whenever $k > C$; otherwise the part was rejected and discarded. The second was a probabilistic admission controller that admitted a part that had a deadline $k \cdot \theta_{\text{min}}$ with a probability

$$\frac{1}{1 + e^{-a(k/C-1)}}$$

where $a = 4.0$ was a constant.

The performance of these two admission controllers is shown in Figure 5.2. The top line shows the percentage of parts that do not meet their deadline when no admission controller is used. The two lines on the bottom of the left part show the percentage of parts lost with the simple deterministic and probabilistic admission controllers. As expected the percentage of parts lost for the probabilistic controller is marginally larger than that of the deterministic controller.

The histogram shown in Figure 5.3 was obtained when parts were injected at a constant rate of 0.8 at $I_3$ and at Poisson(0.4) at the other inputs. Parts at $I_3$ were assigned an integer value of $k$ that was drawn uniformly from the range $[1, 7]$. 

34
Figure 5.2: There is a steep increase in the number of parts that did not meet their deadline around the converged value of $C$ shown in Figure 5.1.

The probabilistic admission controller was used at $I_3$. The converged value of $C$ in this case was 4.86. The histogram shows the number of parts for each value of $k$ that were admitted. The deterministic admission controller would have not admitted any of the parts with value of $k \leq 4$. The probabilistic controller allowed some of these parts; however, some parts with value of $k = 1$ or 2 did not meet their deadline. However, over 9000 parts that had $k < 5$ were admitted to the system.

5.3 Discussion

Admission control is necessary when different parts have different deadlines and the system must guarantee that the part will be delivered at the output before the deadline has elapsed. Then, instead of controlling rate of injection, it is necessary to make
a separate decision on admission of each part based on its deadline. Once again, since there may not be a model or all the details of the system, adaptive mechanisms are interesting options. The results in this chapter based on the simple admission controller highlight the need for more capable admission controllers to be designed in the future.

Adaptively re-routing parts based on noisy feedback about the current performance of the system is an effective response to handle failures in some parts of the system. Re-routing is also necessary to enable topology reconfiguration or adapt the QoS to changes in the types of parts that arrive on the system.
CHAPTER VI

CONCLUSIONS

This thesis presented Learning Automata (LA) based controllers for input-rate control in composable and reconfigurable conveyor systems. These controllers are adaptive mechanisms that learn to choose the optimal rate based on noisy feedback from the system. The feedback provided was the average of recent end-to-end latencies and throughput that was observed in the system. The effectiveness of these controllers were demonstrated through a series of simulation experiments using a representative conveyor system that captured most of the challenges presented by this class of systems. The simulation results showed that the simple stochastic algorithm used by the controller was capable of selecting the best action. Further, the controller automatically responded to changes in system configuration by changing the input-rate suitably. It is indeed interesting that this adaptation comes about without the controller needing any extra information about the state of the system, such as a failure having occurred.

The rate of convergence of the learning controllers as seen from the simulation experiments was not really satisfactory. In all cases, the probability of the optimal action did not become close to unity when the simulation was stopped. Nevertheless, the controller was still effective in delivering reasonably good performance. The
issue of increasing the rate of convergence of learning must be addressed in a future
investigation that extends this current work.

The central idea behind these simple on-line controllers, namely, utilizing
stochastic feedback from the environment to learn the correct action over time, is
intriguing. The effectiveness of the input-rate controllers and the performance of the
probabilistic admission controller are encouraging and motivates future investigations
to design and validate similar LA based controllers for route control and admission
control.
BIBLIOGRAPHY


