A Reinforcement Learning Approach for Production Control in Manufacturing Systems

A.S. Xanthopoulos, D.E Koulouriotis[1] and A. Gasteratos

Democritus University of Thrace, Hellas, email[1]: jimk@pme.duth.gr

Abstract. The problem of production control in serial manufacturing lines that consist of a number of unreliable machines linked with intermediate buffers is addressed. We make use of Reinforcement Learning methodologies in order to derive efficient control policies. Our aim is to derive control policies that are more state-dependent and therefore more efficient than well-known pull type control policies such as Kanban. Manufacturing systems of this type are studied under average measures such as average WorkInProcess inventories etc. and thus, a learning algorithm from the currently developing field of Average Reward Reinforcement Learning was applied. The Reinforcement Learning control policy was compared to three existing efficient pull type control policies, namely Kanban, Base Stock and CONWIP on the basis of simulated data and found to outperform them. The simulation experiments involved a single-product system with two machines that allows backordering. Numerical results are presented along with a qualitative interpretation of our findings. The paper concludes with directions for future research.

1. INTRODUCTION

The manufacturing process of a plethora of industrial end products is structured as a series of operations such as machining, forming etc. that sequentially take place on raw parts and sub-assemblies in order to form the finished product. The various manufacturing stations of the system that can be single machines, manufacturing cells etc. have non-deterministic production times and are susceptible to failure. The unreliability of the manufacturing operations along with the stochastic demand for final products dictates the use of safety buffers of intermediate and finished parts in order to attain the target service rate. However the use of safety stocks incurs significant holding costs that could bring the manufacturer to a position of competitive disadvantage and this is why a lot of effort has been put on the implementation of efficient methods for coordinating the manufacturing process.

A time-honored approach to modeling serial manufacturing lines is to treat them as Markov Processes (Gershwin, 1994, Veatch and Wein, 1992). The sequential decision-making task of coordinating production control can then be formulated as a Markov Decision Problem (see Puterman, 1994 and Littman et al, 1995). The solution to a Markov Decision Problem, (MDP), is a mapping from states to actions which is called optimal policy, that determines state transitions to maximize/minimize a properly defined performance criterion. There are well-known iterative algorithms for solving MDPs such as policy iteration (Howard, 1960), value iteration (Bellman, 1957) and variations of the two above mentioned techniques (asynchronous dynamic programming algorithms in Bertsekas, 1987). However the classic dynamic programming approach entails two major drawbacks: Bellman's famous curse of dimensionality and the need for a complete mathematical model of the underlying problem. The term dimensionality curse refers to the exponential computational explosion that takes place with the increase of the system state space rendering a wide range of realistic problems intractable by dynamic programming, (DP), algorithms. Another serious limitation posed by DP techniques is that they require transition probability and reward information in order to be employed in a certain problem. The later severely complicates the task of system modeling while in the same time rules out real-world problems where a complete model is simply not available.

Due to the above mentioned limitations, researchers have focused their efforts at developing sub-optimal yet efficient production control policies. A class of well-studied control mechanisms known as pull type control policies/mechanisms has come to be widely recognized as capable of achieving quite satisfactory results in serial manufacturing line management. These policies are the result of the emergence of the JustInTime/lean manufacturing philosophy and their main representative is the Kanban control policy that was originally developed by the Toyota Motor industry and became the topic of considerable research thereafter, (Sugimori et al, 1977, Buzacott and Shanthikumar, 1993, Berkley, 1992, Karaesmen and Dallery, 2000). Two other important and also well-studied pull control policies are the Base Stock (Buzacott and Shanthikumar, 1993) and the CONWIP (Spearman et al, 1990) control policies.

Pull type manufacturing control mechanisms as efficient as they can be, still remain sub-optimal heuristics with a number of serious inherent weaknesses, (e.g. the Kanban policy has limited flexibility when it comes to rapid reaction to incoming demands, the WorkInProcess inventory is unbounded in a Base Stock system), and require a lot of fine-tuning in order to perform well. Another important shortcoming is that pull control mechanisms do not take into consideration the operational status of the system's machines, that is whether a machine is in working condition or has failed and is currently under repair.

The writers' motivation was to utilize artificial intelligence methodologies in order to derive automatically dynamic, more state-dependent, near-optimal policies that would outperform the existing pull type policies. In our attempt to do so, we use a method from the field of Reinforcement Learning (Sutton and Barto, 1998, Kaelbling et al 1996). Related work can be found in Paternina-Arboleda and Das (2001) and Mahadevan and Theocharous (1998) but their learning algorithm is different than the one used here. Reinforcement Learning, (RL), can be viewed as a collection of techniques and learning algorithms for teaching agents optimal policies through interaction with their environment which is usually simulated by a computer program. Reinforcement Learning has received a lot of attention over the last few years as it provides the means for solving many challenging problems, previously considered to be intractable. In this paper we applied a variation of R-learning (Schwartz, 1993), presented in Singh (1994) which uses average reward as its performance criterion, a choice that was dictated by the fact that we are interested in the average performance measures of the manufacturing system such as average WorkInProcess inventories, (WIP), etc. Average Reward Reinforcement Learning is still considered to be at an early stage with many important issues remaining open to discussion whereas RL algorithms that maximize average reward currently existing in the open literature are scarce. The first Average Reward RL algorithm was R-learning (Schwartz, 1993) followed by some variations of the original R-learning algorithm due to Singh (1994). Model-based average reward RL algorithms were developed by Mahadevan (1996) and Tadepalli and Ok (1998), Das et al. (1999) introduced SMART, an average reward algorithm for semi-Markov systems while Gosavi (2004) presented an RL algorithm for maximizing average reward based on policy iteration.

The RL control policy that we obtained by applying Singh's algorithm was compared to three existing efficient pull type control policies, Kanban, Base Stock and CONWIP on the basis of simulated data and found to outperform them in the task of minimizing WIP subject to a Service Level, (*SL*), constraint. The simulation experiments involved a single-product system with two machines that allows backordering. Analytical numerical results are presented along with a qualitative interpretation of our findings.

The remaining material of this paper is structured as follows. In Section 2 we offer the system description of a serial manufacturing line. Sections 3-3.3 are devoted to the presentation of the Kanban, Base Stock and CONWIP control policies. In Sections 4-4.2 we discuss the main aspects of the Reinforcement Learning framework and the algorithm that we applied in the underlying problem. We report our findings from the simulation experiments that we conducted for a two-station serial line in Sections 5-5.3. Finally, in section 6 we state our concluding remarks and point to possible directions for future research.

2. SYSTEM DESCRIPTION

In this paper we examine manufacturing systems that produce end products of a single type and consist of several manufacturing stations in series. A manufacturing station is a manufacturing/inventory module that incorporates one or more production operations (e.g. a single machine, a flexible manufacturing cell etc) grouped together and labeled as the manufacturing facility and a physical area of storage, the output buffer of that station. Production in batches is not allowed in this system as well as the reworking of parts in the same manufacturing station. The case of defective station *i* parts is also ruled out. Raw parts undergo production operations in the various manufacturing stations sequentially and are gradually converted into final products as they follow their downstream trajectory. By assumption, the system has an infinite supply of raw materials, so the first manufacturing facility is never starved. Demands arrive to the system at random time intervals and request the release of one part from the finished goods buffer. Provided that there is at least one finished part in the last buffer, the customer demand is satisfied instantaneously. If there are no final parts available, the demand is backordered. In the case where the system operates under a complete backordering policy and since customer impatience is considered to be absent in our model, no customer demand is lost to the system. Another option is the partial backordering policy where backorders are not allowed to exceed a predefined level. Manufacturing facilities are capable of processing only one part at a time. As soon as a station *i* part is produced, it is placed in the *i*th output buffer of the serial line where it waits for two conditions to be satisfied in order to be released to the next manufacturing facility: availability of the station *i* manufacturing station and authorization by the control policy. There is no delay in material handling between stations. Production control policies provide alternative ways of coordinating the release of parts from one station to another. Note that the points where production control is exerted by the controller are not fixed but depend on the control policy under which the system operates, e.g. one extreme condition is when production control is applied only to the first station as we will see in a subsequent section. All manufacturing facilities have random production time, time between failures and repair time. Time intervals between demand arrivals are also stochastic.

3. PULL PRODUCTION CONTROL POLICIES

Production control schemes that coordinate the production activities in a serial line based only on actual occurrences of demand are classified as pull type production control policies/mechanisms. Pull type control policies implement the JustInTime, (JIT), manufacturing philosophy and have attracted considerable attention over the past years as they are widely considered to outperform MRP-based production control systems. According to the JIT manufacturing philosophy a manufacturing system should maintain the minimum levels of safety stocks that are required in order to meet the target service level while in the same time has the ability to react rapidly to incoming orders. Pull production control policies are efficient heuristics with the major advantage of implementation simplicity that characterizes the host of them. The most important issue when applying pull type control of a certain type in a production system is to determine the best policy within this class of policies by appropriate control parameter selection. In sections 2.2 through 2.4 we present three fundamental pull control systems and namely, the Kanban (Sugimori et al, 1977), the Base Stock (Buzacott and Shanthikumar, 1993) and finally the CONWIP (Spearman et al, 1990) control mechanisms.

3.1 Kanban Control Policy

The Kanban production control policy was originally developed by the Toyota Motor Company in the mid-seventies and turned out to become almost a synonym for JIT manufacturing in the years to follow as it provided a conceptually clear, easy to implement and efficient way to coordinate the production process in a serial line. The philosophy of the Kanban control mechanism is quite simple. Each manufacturing station in a Kanban system has a fixed number of station *i* production authorizations; the system's control parameters which are set at design time. The K_i production authorizations (or kanbans) equal the maximum number of parts that are allowed in station *i*. Whenever a station *i* part exits its corresponding output buffer the controller authorizes the release of a station *i*-1 part into the next manufacturing facility. Thanks to this simple policy Kanban control achieves very tight coordination of the several production stations appearing in a system. However this happens in the expense of the manufacturing system's ability to respond swiftly to customer demands as the information of an arrival is transmitted across the system station by station through kanban authorizations. If there is a point where a part is not available in order to be released to the next station, then no Kanban card is sent to the upstream production facility and the transmission of the information of the customer order is interrupted.

3.2 Base Stock Control Policy

The term base stock, also written as installation stock, is borrowed from inventory systems theory (Axsäter and Rosling, 1993), and refers to the initial levels of the systems intermediate and final product buffers. The initial base stock levels B_i are the system's only control parameters as the WIP is not constrained as in the Kanban control mechanism by a finite number of local production authorizations. The term WIP refers to the total number of parts that exist in the buffers of the manufacturing system. "Station *i* WIP" refers to the number of parts that are in the output buffer of this station. On the contrary, the WIP in a Base Stock system is not bounded and that is why this control policy can never be optimal as it was proven in Veatch and Wein (1994). The arrival of a demand is immediately transmitted to every manufacturing station in the system authorizing it to produce a new part. The theoretically strong point of this control mechanism is the high degree of responsiveness to the demand's fluctuations with the disadvantage of insufficient inventory control.

3.3 CONWIP Control Policy

CONWIP control was introduced in 1990 by Spearman et al. as a novel control mechanism for lean manufacturing, however as argued by Liberopoulos and Dallery (2000), it can be easily classified as a special case of the Kanban control system. In a CONWIP system production control is applied only to the first manufacturing station, whereas all the remaining stations have the perpetual authorization to produce provided that this is feasible. The CONWIP controller authorizes the first station to fabricate a new station 1 part as soon as a part exits the finished goods buffer in an effort to maintain the total WIP constant (CONstantWIP). An inherent characteristic of this control mechanism is that the WIP tends to accumulate in the last buffer. The system's sole control parameter is the maximum number of parts allowed in the last buffer or equivalently the number of CONWIP type production authorizations. A CONWIP manufacturing system can be seen as a one-station Kanban system where all of the serial line's production activities have been functionally aggregated in a single manufacturing facility.

4. FUNDAMENTALS OF REINFORCEMENT LEARNING

The majority of the published research in Reinforcement Learning is devoted to the following two models of optimal agent behavior; the maximization of the cumulative sum of rewards and the maximization of the discounted sum of rewards. The first case is appropriate for tasks that are naturally sub-divided in separate episodes while the second measure is used as a means to keep the infinite sum of rewards bounded in continual tasks where the decision-making process is repeated forever. Discounting future rewards makes perfect sense when it comes to problems encountered in e.g., economy, however this optimality criterion is not well-tailored for numerous Reinforcement Learning tasks. Many problems from queuing theory, manufacturing and other cyclical tasks are studied under average measures and therefore it is more logical to want to maximize the average reward per time step collected by the agent:

$$\lim_{n \to \infty} E\left(\frac{1}{n}\sum_{t=1}^{n} r_t\right) \tag{1}$$

A policy π^* that maximizes average payoff per time step is called gain-optimal. There are known issues concerning this optimality framework that can be alleviated by adopting a more generalized optimality criterion known as bias-optimality, (Kaelbling et al., 1996, Mahadevan, 1996) that also takes into consideration the reward gained in the initial phase of the agent's life. However for most practical purposes it is adequate to use the gain-optimal model in addition to the fact that bias-optimal algorithms are still considered to be experimental and significantly less well-understood.

A stationary policy *π*, is a mapping from states, $s \in S$, and actions $a \in A(s)$, to the probability $prob(s,a)$ of choosing action a when in state *s*, where *S* is the finite set of states in the task and *A(s)* the permissible actions in state *s*. A policy in general may be non-deterministic. Model-free Average Reward Reinforcement Learning algorithms are iterative stochastic approximation algorithms that use sample state transitions and sample rewards generated by simulation models in order to estimate relative value functions and average payoff. Informally, relative value functions are functions of states (or state action pairs) to real values that quantify the ''usefulness'' of being in a given state (or taking a certain action when being in a given state) when following a particular policy. Here, the term usefulness refers to the expected future rewards of the agent. Let ρ^π denote the average expected reward per time step under policy π:

$$\rho^\pi = \lim_{n \to \infty} E\left(\frac{1}{n}\sum_{t=1}^{n} r_t\right) \tag{2}$$

We define the relative value of state *s* under policy *π* as:

$$V^\pi(s) = E_\pi\left(\sum_{n=1}^{\infty} r_{t+n} - \rho^\pi \mid s_t = s\right) \tag{3}$$

Similarly, the relative value of taking action a in state *s*, or state-action value for short, is written as:

$$Q^\pi(s,a) = E_\pi\left(\sum_{n=1}^{\infty} r_{t+n} - \rho^\pi \mid s_t = s, a_t = a\right) \tag{4}$$

The solutions to the Bellman equation, (Equation 5), for average reward MDPs are the average payoff and the relative state-action values for the optimal policy π^*, denoted by ρ^* and Q^* respectively:

$$Q^*(s,a) = r(s,a) - \rho^* + \sum_{s' \in S} prob_{ss'}(a)\left[\max_{a' \in A(s')} Q^*(s',a')\right],$$
$$\forall s \in S, a \in A \qquad (6)$$

where $prob_{ss'}(a)$ is the transition probability from state *s* to state *s'* when choosing action *a*, and $r(s,a)$ the reward on executing action *a* in state *s*.

4.2 Variation of R-Learning

The first average reward RL algorithm to appear in the literature was Schwartz's R-learning (1993), and since then only a few more algorithms were added to this category, (see Introduction for a listing of relevant algorithms). In this paper we used a variant of R-learning developed by Singh (1994) in order to derive efficient control policies for serial manufacturing lines. This improved version of R-learning is presented in the following Listing 1.

Listing 1. Variant of R-learning (Algorithm 3)

```
t=0, initialize $\rho_t$ and $Q_t(s,a)$ for all s, a.
  • Let s be the current state. Select $a \in A(s)$ according to action selection strategy, (e.g. e-greedy)
  • Observe immediate reward $r_{t+1}$ and next state $s'$
  • $Q_{t+1}(s,a) = (1-\alpha_t)Q_t(s,a) + \alpha_t[r_{t+1} - \rho_t + \max_{a'} Q_t(s',a')]$, $a' \in A(s')$
  • $\rho_{t+1} = (1-\beta_t)\rho_t + \beta_t[r_{t+1} + \max_{a'} Q_t(s',a') - Q_t(s,a)]$, $a' \in A(s')$
  • Decrease parameters α and β, and the exploration parameter e
  • Loop
```

ρ_t and $Q_t(s,a)$ are approximations of the average reward and the state-action values, respectively, in epoch *t*. $A(s)$ is the set of admissible actions in state *s*. α_t is the learning rate for the Q-values and β_t is the learning rate for the average reward estimate, in decision making epoch *t*. *e* is the exploration parameter.

4.3 Methodology and Notation

In this section we discuss the way we fit the problem of coordinating production in a serial line into the Reinforcement Learning framework. First of all we must point out that there is a subtle distinction that needs to be drawn between the environment's relative position to the agent in the RL formulation and in the real world. In an actual manufacturing line the controller is embedded to the system whereas in the RL model the decision-making agent and the controlled system are two distinct entities interfaced together. The discrete set of states that the agent can find itself in is denoted by S and is given by the following expression:

$$S = \{(M_i, O_i, Bck) : i = 1,2,....,n\} \qquad (7)$$

where *n* is the number of manufacturing stations and $M_i \in \{1,0,2\}$ are the possible conditions of the station *i* manufacturing facility. We denote failure in the manufacturing facility by 0, working condition by 1 and the state of being idle by 2. $O_i \in \{0,1,2,....\}$ is the station *i* buffer's state and finally, $Bck \in \{0,1,2,....\}$ is the number of backordered demands. The available actions to the controller are *authorize production* and *do not authorize production* of a new station *i* part for each of the *n* manufacturing stations, hence the number of admissible actions is 2^n. Let *A* denote the action space; as a result the complete state-action pair space is written as $S \times A$. The decision-maker interacts with the controlled system at discrete time steps $t = 1,2,3,...$ where the agent receives a representation of the environmental state and based on that information selects an action.

Each decision made by the agent has an associated cost which is given by Equation 7:

$$r_{t'} = -\sum_{i=1}^{n} q_i \overline{O_i} - c\overline{Bck} \qquad (8)$$

where $\overline{O_i}$ and $\overline{Bck}$ are the time-averaged WIP in buffer *i* and level of backordered demands respectively until the next decision making epoch while q_i and c are positive constants. An action that yields relatively large amount of reward (or to be more precise low cost in our model) causes an improvement on the corresponding relative action value. In the algorithm that we implemented for the purposes of this paper state-action pair values are stored in a look-up table but in order to cope with large scale problems the use of some kind of a function approximation would be imperative. At each discrete time step *t* the agent implements a policy π_t on the basis of the relative Q-values of the state-action pairs. Most of the time the controller acts *greedily*, that is, it selects the action with the higher Q-value but in a fraction *e* of time it deviates from this behavior by selecting actions randomly in order to explore actively the state-action space. The agent explores intensively in the initial phase of the control task and then the amount of exploration slowly decays with time in order to allow convergence to the derived as optimal policy. The agent's goal which is conveyed to it through the cost assignment scheme as we will elaborate in the section to follow is to minimize WIP inventories subject to a service level constrain:

$$\min \sum_{i=1}^{n} \overline{WIP_i} \qquad (9)$$

under the constraint:

Service Level>*t%*

where *t* is the service level target also known as the *fill rate* and specifies the proportion of customer orders which are satisfied instantaneously from the existing finished products stock.

5. SIMULATION

5.1 Simulation Case

The simulation experiments involved a two – station manufacturing line with equal operation times that operates under a policy that allows backorders. Machines operate with service rates which are normally distributed random variables with mean 1.0 parts/time unit and s.d. 0.01 ($R_p \sim N(1.0,0.1)$). Repair to failure times are exponentially distributed with mean 1000 time units. Failures are operation dependent. Repair times are also assumed exponential with a MTTR of 10 time units. Times between two successive customer arrivals are exponential random variables with mean 1.66 time units. Simulation time is set to 30000.0 time units. The objective is to minimize WIP while maintaining a 90% service level.

5.2 Experimental Setup

Prior to comparing the RL derived policy to the pull type heuristics we need to determine the best control parameters for each one of the Kanban, Base Stock and CONWIP policies. The parameters of the Kanban policy are the number of local production authorizations denoted by K_i while the Base Stock policy is characterized by the initial base stocks denoted by B_i. The parameters of a CONWIP system are the *C* global, (CONWIP type), production authorizations and the initial stock of intermediate and final parts. Note that by increasing the number of kanbans, base stocks or CONWIP type authorizations, the average throughput of the serial line increases and hence, the service level, but so does the WIP along with the related costs. In order to find the best parameters for each pull control policy we perform an incremental search over the space of feasible parameter sets. The results are displayed in Table 1.

Table 1. Pull production control policies parameter sets

	K_1/B_1	K_2/B_2	C
Kanban	1	4	-
Base Stock	0	5	-
CONWIP	2	3	5

The actions available to the agent are to authorize or not production in each one of the system's stations, therefore: *A*={[1 1],[1 0],[0 1],[0 0]}. In order to constrain the number of the system states we define the permissible buffer conditions to be $O_i \in \{0,1,2,3,4\}$ and we allow only partial backordering: $Bck \in \{0,1,2,3,4\}$. The default course of action when the inventory in a buffer rises beyond the allowable limit is to stop production in that certain station. The opposite happens when the number of backordered demands exceeds the predefined levels given that it does not violate the previous rule. The upper limit for the buffer levels is analogous to the control parameters of the pull type policies and the maximum number of unsatisfied demands was set also on the basis of the pull heuristics performance. We should point out that these situations where the control is taken away from the RL agent in fact are very unlikely if impossible to occur in practice due to the cost assignment scheme and the simulation model's parameters but there needs to be a way to handle them for consistency reasons. The complete state-action space then consists of 4(actions)×3^2(machine states) ×5^2(buffer states) ×5(bck states) = 4500 Q-values. As we have mentioned in former sections the only feedback available to the decision-maker is the reinforcement signal. It is essential that this signal has all the necessary information in order to communicate to the agent what we expect it to do. If we used Eq. 9 as the cost function, the agent would try to minimize the average WIP with no constraints whatsoever. The agent would minimize WIP easily simply by never giving the order to produce a single part. Our goal is to minimize WIP under the constraint of maintaining the service level above a specified target value. This is achieved by finding the appropriate values for parameters $q_i, i=1,2$ and *c* from Equation 7. Parameters $q_i, i=1,2$ can be interpreted as the cost of storing one part in output buffer *i* per time unit and *c* as the cost of unsatisfied demand per time unit. For example, if we increase parameter q_2, the resulting policy will yield lower WIP in buffer 2 at the expense of a lower service level. After a lengthy trial-and-error procedure we came up with $q_1 = q_2 = 10.0$ and $c = 100.0$ as the most suitable parameters for this task.

5.3 Results-Discussion

This section is devoted to the presentation and analysis of the numerical results that were produced via experimentation with simulation models. It is known from Reinforcement Learning theory that in order for the agent to converge to the true state-action values and average reward, all states and actions must be visited infinitely many times. Luckily, for most practical purposes a large amount of exploration will do the job, that way avoiding exhaustively long simulation times. Note that a key feature that renders RL superior to other intelligent methodologies is that it does not spend much time on evaluating state-actions with low probabilities of occurrence but focuses at maintaining reliable estimates for the Q-values of the more probable state-action pairs. In order to ensure convergence stability we set the decay factors of the learning rates to very low values. The RL algorithm was executed for 20 times the simulation time which was used to evaluate the performance of the pull type production control policies, that is, $20 \times 30000 = 600000$ time units. Given that the mean time between two successive customer demands was set to 1.66 time units (see section 5.1), approximately 360000 demands arrived to the system in this simulation time. All of these demands were ultimately satisfied by the system so there were, roughly, another 2(machines) × 360000(parts)=720000 "production in machine *i*" simulation events. If we take into consideration the events "failure in machine *i*" and "repair in machine *i*" the number of simulation events/decision making epochs is well beyond one million. This simulation time allows all state-action pairs to be visited many times and thus, calculate reliable Q-values estimates. Figure 1 shows the convergence of the RL algorithm. We set $Q_0(s,a) = 0.0, \; \forall s,a$. This implies that all admissible actions in all states have initially equal probabilities to be selected by the controller, including the optimal ones. As the number of iterations advances, the algorithm actually concentrates its choices to the perceived as optimal ones but this might not be clearly depicted in the average reward plot (Figure 1) due to the inaccuracy of the average reward estimate in the initial phase. Figure 1 illustrates the average reward estimate calculated by the RL algorithm and not the actual average reward. Note that in Singh's algorithm, the average reward estimate is calculated with the help of the immediate reward and the state-action values $\max_{a'} Q_t(s',a')$ and $Q_t(s,a)$ (see Listing 1).

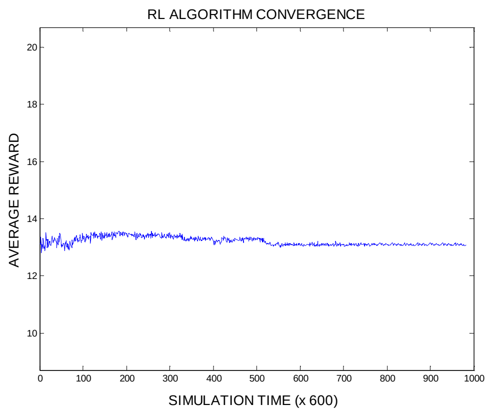

Figure 1. Average reward of the RL control policy

The initial heavy exploration phase is depicted in the fluctuations of the curve in Figure 1, while in the right-most part of the plot where the learning curve forms a straight line we can see the convergence phase of the algorithm. At this point we should mention that if the learning rates are prevented from decaying to zero the algorithm does not converge completely and is rendered suitable for a non-stationary problem where, i.e. the demand generating process slowly changes with time, a situation which is impossible to handle for the pull heuristics. The average WIP distribution through out the system and the backordered demands average together with the service rate that was achieved by the steady RL policy as well the ones of the three pull type control policies are presented in Table 2.

Table 2. Average buffers levels-backorders averages-service levels

	$\overline{O_1}$	$\overline{O_2}$	$\overline{Bck}$	*SL*
Kanban	0.39	2.88	0.18	93.12%
Base Stock	0.11	3.24	0.17	93.68%
CONWIP	0.05	3.28	0.17	93.70%
RL	0.09	3.11	0.18	93.10%

The simulated data are presented graphically in Figure 2. For all policies we reported service levels of approximately 93% and an average of about 0.18 unsatisfied customer orders. The characteristic attribute of the Kanban control policy which is tight coordination between the various manufacturing stations of the system is evident in the first row of Table 2. Indeed, the Kanban system displays the most uniform WIP distribution among all pull type policies but also it maintains the highest WIP inventory in buffer 1 during the system's operation. The Base Stock mechanism offers a significant improvement over the Kanban policy in terms of WIP accumulation in the first buffer thanks to the initially zero base stock in that buffer but this happens in the expense of WIP build-up in the finished goods storing facility. In a system that operates under the CONWIP policy all the machines except the first one have the perpetual authorization to produce whenever they can, and this is why WIP tends to accumulate in the last buffer.

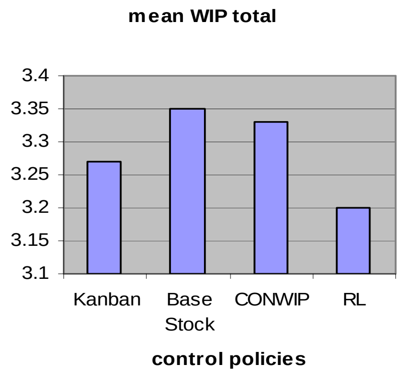

Figure 2. Total WIP

This situation is depicted in Table 2 where we observe that the average finished products inventory in the CONWIP system is the highest of all control policies. As a consequence, parts are not stalled in intermediate buffers due to e.g. lack of production authorization which in turn results in very low average WIP levels in the first buffer. The RL policy produced slightly higher WIP levels in the first buffer than the best policy found in this part which was CONWIP. It also ranked second in constraining WIP in the second buffer too. However the policy derived using the Reinforcement Learning algorithm outperforms the other three policies when it comes to total WIP as we can see in Figure 2. We could argue that the RL policy combines the desirable aspects of the various pull type control mechanisms while omitting their weaknesses. In order to gain insight of the behavior of the RL policy we examined the Q-values of state-action pairs that we were particularly interested in. It is obvious that the RL policy is considerably more complicated than the pull heuristics as it explicitly maps several hundreds of possible system states into actions and thus not as conceptually clear as these policies but this is not necessary a drawback in the modern automated industrial reality. The reasons why the RL policy outperforms the existing control policies can be sought in the fact that it is more state-aware than classical pull type mechanisms. None of the Kanban, Base Stock and CONWIP policies takes into consideration the events of machine break down and repair and this leads to sub-optimal behavior in many cases. For instance, consider the case where the second production facility is under repair. The CONWIP and Base Stock control policies will keep authorizing the first station to produce new parts regardless of the fact that these cannot be released to the downstream station thus, leading to excessive inventory accumulation in the first buffer. On the other hand, the Kanban control system will not authorize production in a station unless there is a free kanban authorization available, a condition that severely impedes the system's ability to react to customer arrivals. It is true that the RL policy may not behave optimally for numerous rarely visited states but the overall improvement offered to the system performance is more than evident.

6. CONCLUSIONS AND FUTURE RESEARCH

In this paper we use artificial intelligence methodologies to derive efficient, near-optimal control policies for serial manufacturing lines. For this purpose we apply an algorithm from the field of Average Reward Reinforcement Learning. After a brief introduction to the central concepts in Reinforcement Learning and a description of the learning algorithm we proceed to a detailed presentation of the course of action that we followed in order to formulate the problem of production control as a Reinforcement Learning task. We evaluated the performance of the derived control policy through simulation and compared it with those of well-known pull type production control policies. Our experiments involved a manufacturing system with two unreliable machines in tandem where backordering is allowed and a stochastic demand generating process. The competing policies were evaluated on the basis of minimizing WIP subject to a service level constrain. The RL policy was found to outperform the Kanban, Base Stock and CONWIP production control policies in minimizing total WIP primarily due to the fact that it is more state-dependent than the above mentioned heuristics, as it is pointed out in the analysis accompanying the presentation of the numerical results. We plan to proceed to more extensive experimentation in the future in order to produce additional data as well as to extend the use of the RL methodology in manufacturing systems of a different type, as job-shops etc. The tabular approach that we use in this paper is somehow limiting in reference to the size of the problems that it can be used in. As a consequence, another logical direction for research would be to use some kind of function approximation, i.e. an artificial neural network in order to handle large scale problems

REFERENCES

[1] S. Axsäter and K. Rosling, ‘Installation vs. echelon stock policies for multilevel inventory control’, *Management Science*, **39(10)**, 1274-1279, (1993).

[2] R.E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, 1957.

[3] D. Bertsekas, *Dynamic Programming: deterministic and stochastic models*, Prentice Hall, New York, 1987.

[4] B.J. Berkley, ‘A review of the kanban production control research literature’, *Production and Operations Management*, **1(4)**, 393-411, (1992).

[5] J.A. Buzacott, J.G. Shanthikumar, *Stochastic Models of Manufacturing Systems*, Prentice Hall, New York, 1993

[6] T. Das, A. Gosavi, S. Mahadevan and N. Marchallek, ‘Solving semi-markov decision problems using average reward reinforcement learning’, *Management Science*, **45(4)**, 560-574, (1999).

[7] S.B. Gershwin, *Manufacturing Systems Engineering*, Prentice Hall, New York, 1994.

[8] A. Gosavi, ‘A reinforcement learning algorithm based on policy iteration for average reward: empirical results with yield management and convergence analysis’, *Machine Learning*, **55(1)**, 5-29, (2004).

[9] R. Howard, *Dynamic Programming and Markov Processes*, MIT Press, Cambridge,1996.

[10] L.P. Kaelbling, M.L. Littman, A.W. Moore, ‘Reinforcement Learning: a survey’, *Journal of Artificial Intelligence Research*, **4**, 237-285, (1996).

[11] F. Karaesmen, Y. Dallery, ‘A performance comparison of pull type control mechanisms for multi-stage manufacturing’, *International Journal of Production Economics*, **68**, 59-71, (2000)

[12] G. Liberopoulos and Y. Dallery, ‘A unified framework for pull control mechanisms in multi-stage manufacturing systems’, *Annals of Operations Research*, **93**, 325-355, (2000)

[13] M.L. Littman, T.L. Dean, L.P. Kaelbling, ‘On the complexity of solving MDPs’, *In Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, Montreal, Quebec, Canada, (1995)

[14] S. Mahadevan, ‘Average reward reinforcement learning: foundations, algorithms and empirical results’, *Machine Learning*, **22**, 159-196, (1996).

[15] M.L. Putterman, *Markov Decision Processes*, Wiley Interscience, New York, 1994.

[16] A. Schwartz, ‘A reinforcement learning method for maximizing undiscounted rewards’, *In Proceedings of the Tenth Annual Conference on Machine Learning*, 298-305, (1993)

[17] S.P. Singh, ‘Reinforcement learning algorithms for average-payoff markovian decision processes’, *In Proceedings of the Twelfth National Conference on Artificial Intelligence*, 202-207, (1994)

[18] M.L. Spearman, D.L. Woodruff, W.J. Hopp, ‘CONWIP: a pull alternative to kanban’, *International Journal of Production Research*, **28**, 879-894, (1990).

[19] Y. Sugimori, K. Kusunoki, F. Cho and S. Uchikawa ‘Toyota production system and kanban system materialization of just-in-time and respect-for-humans systems’, *International Journal of Production Research*, **15(6)**, 553-564, (1997).

[20] R.S. Sutton and A.G. Barto, *Reinforcement Learning: an introduction*, MIT Press, Cambridge, MA, 1998.

[21] P. Tadepalli and D. Ok, ‘Model-based average reward reinforcement learning’, *Artificial Intelligence*, **100**, 177-224, (1998).

[22] M.H. Veatch, L.M. Wein, ‘Monotone control of queueing networks’, *Queueing Systems*, **12**, 391-408, (1992).

[23] M.H. Veatch, L.M. Wein, ‘Optimal control of a make-to-stock production system’, *Operations Research*, **42**, 337-350, (1994).

[24] C.D. Paternina-Arboleda, T. K. Das, ‘Inteligent dynamic control policies for serial production lines’, *IIE Transactions*, **33**(1), 65-77, (2001)

[25] S. Mahadevan and G. Theocharous, ‘Optimizing Production Manufacturing using Reinforcement Learning’ , *In Proceedings of the Eleventh International FLAIRS conference* , 372-377, (1998).