Inventory Management in Multi-Product, Multi-Demand Disassembly Line using Reinforcement Learning

Emre Tuncel, Abe Zeid and Sagar Kamarthi
Department of Mechanical and Industrial Engineering
Northeastern University
Boston, Massachusetts 02115, USA

Abstract

Disassembly lines are the best way to disassemble products with similar components in large quantities. A disassembly line consists of a series of workstations. However, unlike their assembly counterpart, disassembly lines are affected by factors such as multiple component demand arrivals and end-of-life (EOL) product arrivals, uncertainty in EOL product and component demand arrivals and varying processing times. Altogether, these factors make a disassembly line very complex and cause demand fluctuations in the inventory levels of sub-assemblies and components in the system. The fluctuations in the inventory levels lead to increased cost of operation and decreased customer satisfaction. This research focuses on addressing inventory management problem in multi-product, multi-demand disassembly line using Reinforcement Learning, specifically Q-learning, which learns from experience to perform very effectively in dynamic environments with stochastic elements.

Keywords
Reinforcement Learning, Q-learning, Neural Networks, Disassembly Line, Inventory Management

1. Introduction and Background

Manufacturers, particularly in the appliance, electronics and computer industries are facing the problem of shortened product lifetime due to technological advancements. Shortened product lifetime is causing the consumers to buy the latest and advanced models of products in use before their functional life is over. Instead of returning the end-of-life (EOL) products for material and component recovery, consumers simply discard them. This attitude has serious environmental repercussions on landfills. Thus, governments of the industrialized countries are now forcing original equipment manufacturers (OEM) to extend their products responsibility. The extended product responsibility includes the collection of EOL products and engaging in operations such as disassembly, recycling and remanufacturing to recover components, recover raw material and bring the EOL products to a like-new state. When the EOL products arrive at a disassembly facility, they go through a series of operations for recovery purposes. Unlike typical assembly line, disassembly systems include complexities such as arrival of multiple demand types, arrival of multiple product types and stochastic demand and product arrival. These complexities are not present in a regular assembly system and they can impede the overall productivity and performance of a disassembly system. In this research we are focusing on a disassembly line with uncertain multi-demand and multi-product arrival. In such a disassembly system, the aforementioned complexities can cause inventory fluctuations which, if left unmanaged, can lead to an increase in the overall cost of maintaining the system and a decrease in the customer satisfaction due to unsatisfied demands. In this paper we seek to maximize the efficiency and profitability of the disassembly system described above by managing the inventory using reinforcement learning (RL) techniques.

The sections in this paper are organized as follows: in section 2 we discuss the literature on inventory management in disassembly systems; in sections 3 we elaborate on the problem of inventory management in disassembly lines and issues tied to addressing the problem; in section 4 we provide the framework for the proposed solution methodology (RL); in section 5 we discuss how the proposed solution methodology is applied to an example problem; and in section 6we discuss the results and present conclusions.

Product/component recovery operations and related problems have been studied thoroughly over the last couple of decades due to increasing environmental and economic concerns. McGovern and Gupta (2011) provide a comprehensive review of problems associated with product/component recovery and disassembly.

In the area of disassembly inventory management, Johar and Gupta (2006) developed a mathematical approach to balancing the inventory generated from a disassembly line depending on the shape and size of the components and the amount of space available in the sorting area of the disassembly line. Udomsawat and Gupta(2003, 2005) presented a Multi-Kanban mechanism (MKBS) for disassembly line with single type products, disassembly line with component discriminating demands, and disassembly line with products having multiple precedence relationships.
There have been many studies related to RL, implementation of RL to problems that involve large state/action spaces and integration of neural networks and RL. Bakker (2007) illustrated backpropagation through long short term memory recurrent neural network model/critic. Qiao, *et al.* (2007) described a neural network for selecting actions in a mobile robot with Q-Learning. Shiraga *et al.*(2001, 2002) described a reinforcement learning algorithm for neural networks with incremental learning ability using a resource allocating network and long term memory. Sun, *et al.* (2008) discussed a RL method for continuous state space by utilizing Elman network. Here, the authors utilized a dynamic Elman network to approximate the Q-value for a state-action pair.

To author's knowledge RL techniques (Q-learning algorithm) have not been applied to address disassembly inventory management problem.

2. Disassembly Problem

In section 1 we mentioned that disassembly lines are ridden with many complexities that do not exist in traditional assembly lines. In an assembly line, base or the core of the product enters the line through the very first workstation. Afterwards, operations are performed on the product at subsequent workstations to produce the final product. The finished product comes off the line through the last workstation. Demand exists for the final product and arrives at a single location, namely the last station. Both product/component and demand flow through the system are convergent. Unlike in an assembly line, EOL products can arrive at any workstation of the disassembly line, depending on the product configuration. Each returned product follows a specific disassembly sequence which usually varies among different types of returned products. The missing components in EOL products make it harder to satisfy incoming demand for components. The demand for components can arrive for any component at any workstation. The product/component and demand flow are divergent. Divergent flow factor coupled with uncertainty in demand and product arrival contributes to the fluctuations in the inventory levels. The generic dynamics of the disassembly system under consideration is illustrated in Figure 1.Depending on the dynamics of the system once a product is pulled from its respective buffer it enters a workstation for processing it and is disassembled into sub-assemblies and components. Components are put into component buffers whereas the sub-assemblies are routed to sub-assembly buffers. The incoming products, sub-assemblies and components routed according to the decision made by RL agent. The important factor here is that if the arriving products go through disassembly operations without any demand arriving for components then the system experiences an inventory explosion.

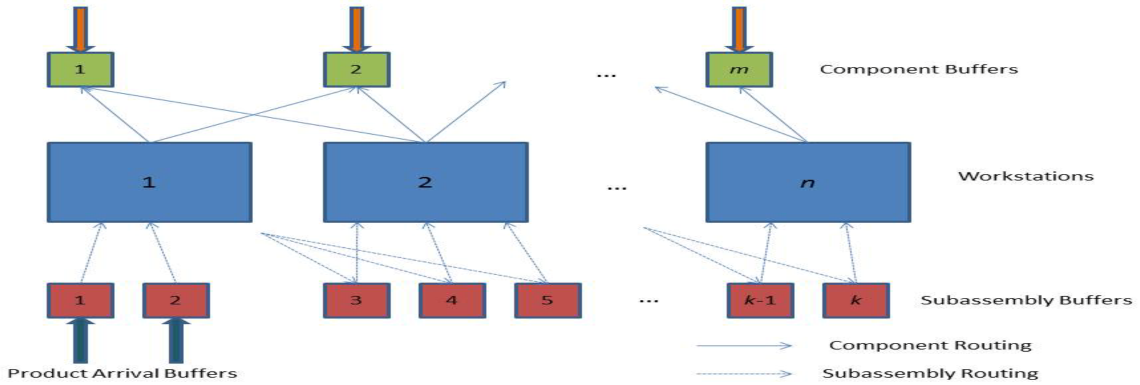

Figure 1: Generic disassembly line representation

In this paper there are two key objectives: To minimize the work-in-process (WIP) inventory and at the same time maximize the number of demands satisfied.

3. Reinforcement Learning Overview

The basic idea of RL is to learn from experience. RL techniques are categorized under "unsupervised learning" branch of the artificial intelligence area in which the decision making agent (DMA) maps situations to actions without needing anything but the reinforcement received from the environment(Watkins, 1989; Watkins and Dayan, 1992; Russel and Norvig, 1995; Sutton and Barto, 1998; Tewari, 2007).At any given time *t* the selects and executes and action based on the current state of the environment. The environment then generates a numerical signal called the reinforcement signal (Sutton and Barto, 1998). It is through this signal that the DMA can understand whether or not it has made a good decision.

RL algorithms are applicable to Markov Decision Processes (MDP) and are structured around clearly defined states, actions and reinforcement. State representation includes the necessary information about the environment and communicates it to the DMA. When the DMA selects and executes an action the state of the environment changes and the DMA receives a numerical reinforcement from the environment based on the desirability of the newly transitioned state. The desirability of the new state is evaluated by the reinforcement signal. Action space contains a set of actions which are available to the DMA at any given time *t*. Suffice it to say that the DMA knows whether or not if it does the "right thing" by perceiving the changes in the states caused by the actions it executes.The RL logic is mapped to values by utilizing several different functions. The mapping allows the DMA to keep track of the long term performance. This way the agent becomes more than a greedy decision making entity which seeks to maximize only the immediate reinforcement received from the environment. In this research we utilize the *Action-Value* (state-action value) function to facilitate the learning. For a set of finite discrete states, $s \in S$, and a set of discrete actions, $a \in A(s)$, the action-value function is denoted as follows:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\} \qquad (1)$$

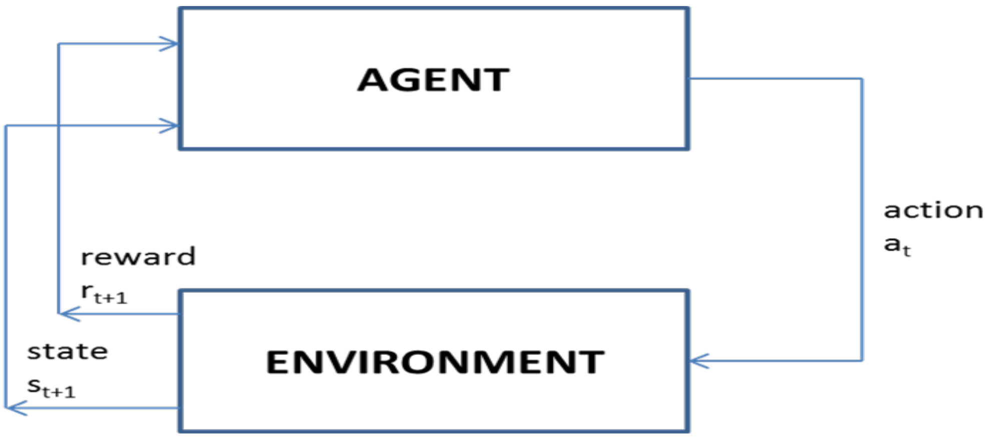

Figure 2: RL framework

Figure 2 illustrates the basic RL dynamics. In Eq. (1) γ denotes the discount rate. Discount rate takes a value in the $0 < \gamma \leq 1$ range; it denotes the present value of rewards received in future states. The function given in Eq. (1) represents the value of taking action *a* in state *s* under a policy π, $Q^{\pi}(s, a)$, denoted as the expected reward received starting from *s*, taking action *a*, and thereafter following policy π. A policy, π, is the probability of selecting $a \in A(s)$ at states $\in S$. RL procedures commonly categorized as *on-policy* or *off-policy* depending on whether or not they strictly follow a policy or not.

4. RL Implementation and Sample Problem

Conventional Q-learning algorithm is not able to grant the necessary "learning" when the DMA has to deal with large state spaces. This is the case with the inventory management problem in disassembly lines which were described in section 3. The size of the state space makes it impossible to use the look-up table to keep track of and update the Q-values. In this research we utilize multilayer neural networks (NN) to overcome this challenge.

Q-learning works by incrementally updating the values of actions in states. Before learning begins the Q-values of the state-action pairs are initialized. Each time the DMA executes an action a from a set of actions A and receives a reward at a particular state s which belongs to state space $S$the state-action value $Q(s_t, a_t)$ is updated following a simple value iteration update which is defined as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(s_t, a_t)[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \qquad (2)$$

In Eq. (2), $s_t$and a_t denotes the state of the system and action executed at time t whereas$r_{t+1}$is the reward given at time $t + 1$, $\alpha_t(s,a)$ $(0 < \alpha \leq 1)$ is the learning rate and γ is the discount factor. However in order to use the value iteration update rule defined in Eq (2) the values of the current and future state-action pairs are required. Due to large state space the look-up table becomes too limiting to use. To overcome this problem,we utilize multilayer networks to store the Q-values. Figure 3 illustrates Q-value approximation using neural networks.

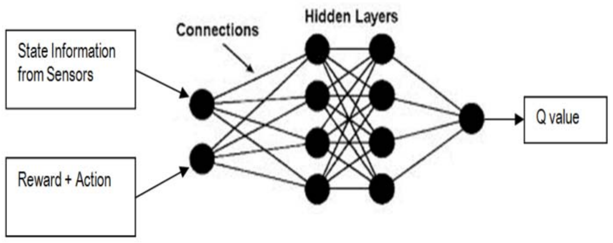

Figure 3: Q-value approximation

The neural network illustrated in Figure 3 consists of 4 layers; an input layer (where information regarding the state, action and reward is received), two hidden layers and an output layer from which the approximated Q-values are obtained.

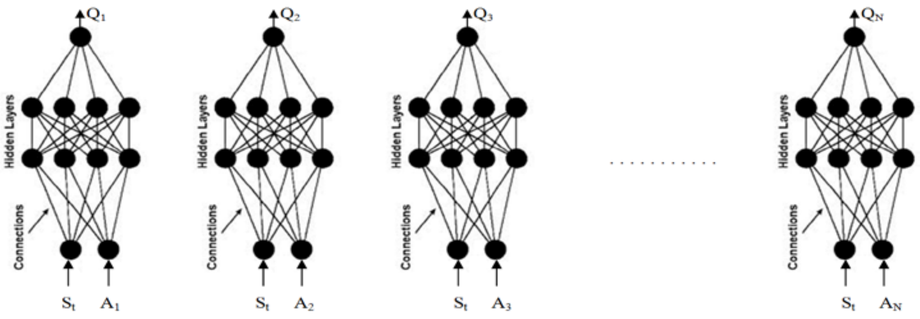

Figure 4: Using multiple NNs for each action

In Figure 4, S_t represents the state of the system at time t, and A_t denotes the action executed at time t. At this time we select the neural network that yields the maximum Q-value. Only the selected neural network gets its weights updated. Afterwards, the state of the system transitions into a new state based on the action that corresponds to the selected neural network. The neural network that yields the max Q-value is also found at this new state in order to obtain $max_a Q(s_{t+1}, a)$. After Q-values for time t and time t+1 are obtained the value iteration update rule defined in Eq. (2) is utilized. Figure 5 illustrates the utilization of neural networks for the resultant state following to execution of an action.

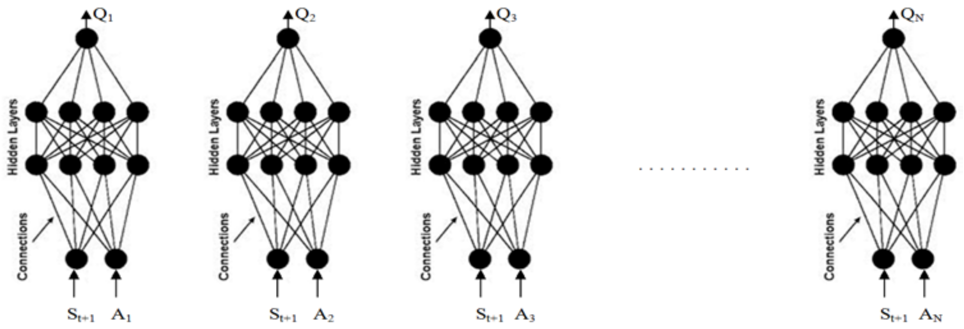

Figure 5: Using NNs for the resultant state

The proposed solution methodology provides the following contributions: (i) it enables us to handle a highly dynamic and uncertain environment without making any unrealistic assumptions; (ii) integration with NN allows us to overcome the curse of dimensionality. We applied the proposed RL methodology in order to test its the efficiency. In the sample problem there are three types of products. Data regarding the products and their disassembly sequences are obtained from Udomsawat and Gupta (2005) and are listed in Table 1.

Table 1: Sequences and arrival patterns of EOL products

Appliance	**Component Structure**	**Disassembly Sequence**	**Mean Arrival Rate(units/hour)**
Microwave Oven	A-B-C-E	1-2-3-4	15
Washer/Dryer	A-B-C-D-E	1-2-3-4	15
Refrigerator	C-B-A	3-2-1	15

Components are processed at different workstations. The demand arrival rate and disassembly processing time for each component is listed in Table 2.

Table 2: Component and operational data

Component Type	**Mean Demand Arrival Rate (units/hour)**	**Mean Disassembly Time (minutes)**
A	10	8
B	10	12
C	10	8
D	10	4
E	10	4

Dynamics of disassembly line under consideration is illustrated in Figure 3. Similar to the generic disassembly line representation presented in section 3, the unnecessary disassembly of components leads to an inventory explosion. We initially benchmarked RL algorithm against traditional push system. Due to aforementioned reasons push system is the worst production strategy that can be applied to the example disassembly line.

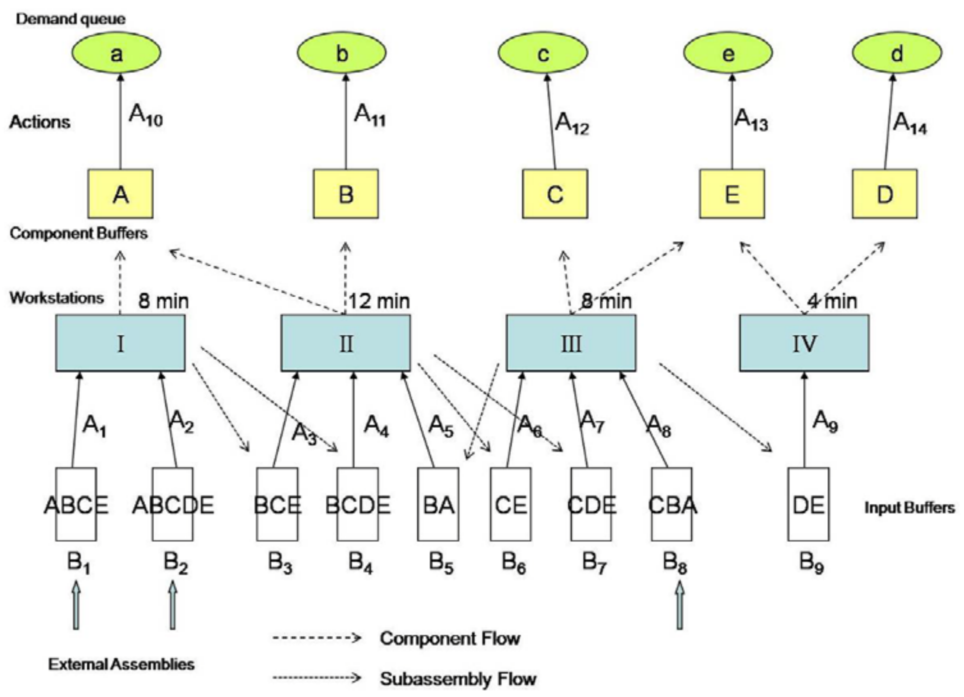

Figure 6: Workflow

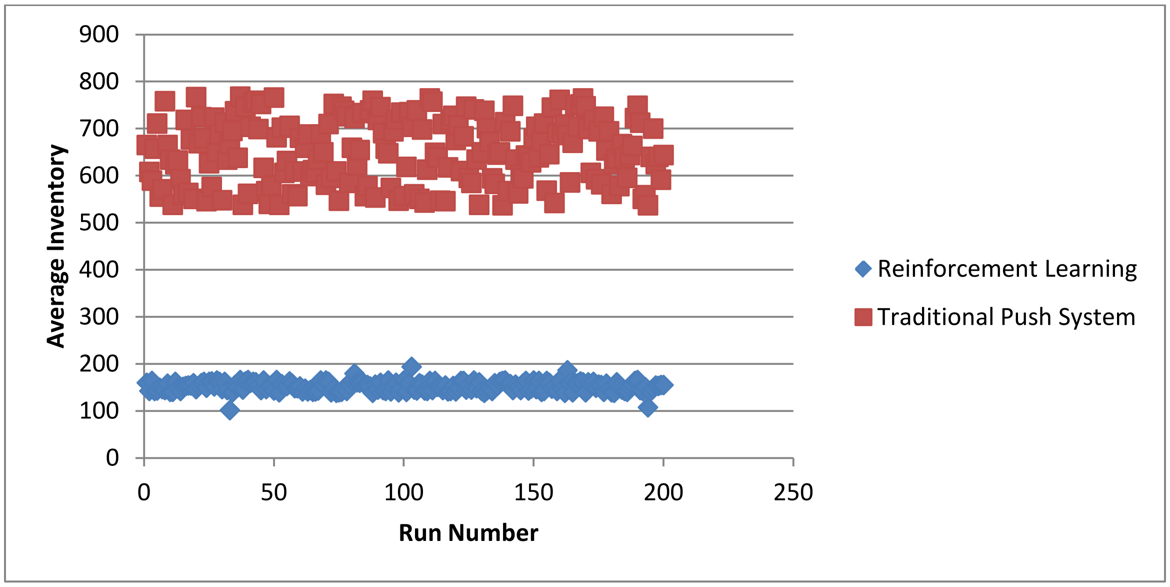

Figure 7: Preliminary Analysis

Figure 7 illustrates the average inventory obtained from 200 independent runs for both RL and traditional push system. We use average inventory as a performance metric because it simply indicator of the overall inventory carried throughout the span of the execution. Our preliminary analysis clearly indicates that RL outperforms the traditional push system. We also analyzed the changes in the inventory accumulation in the system throughout the period of execution in order to gain better understanding about agent's behavior and the decision making process. Figure 8 illustrates the changes in the WIP inventory level vs. time.

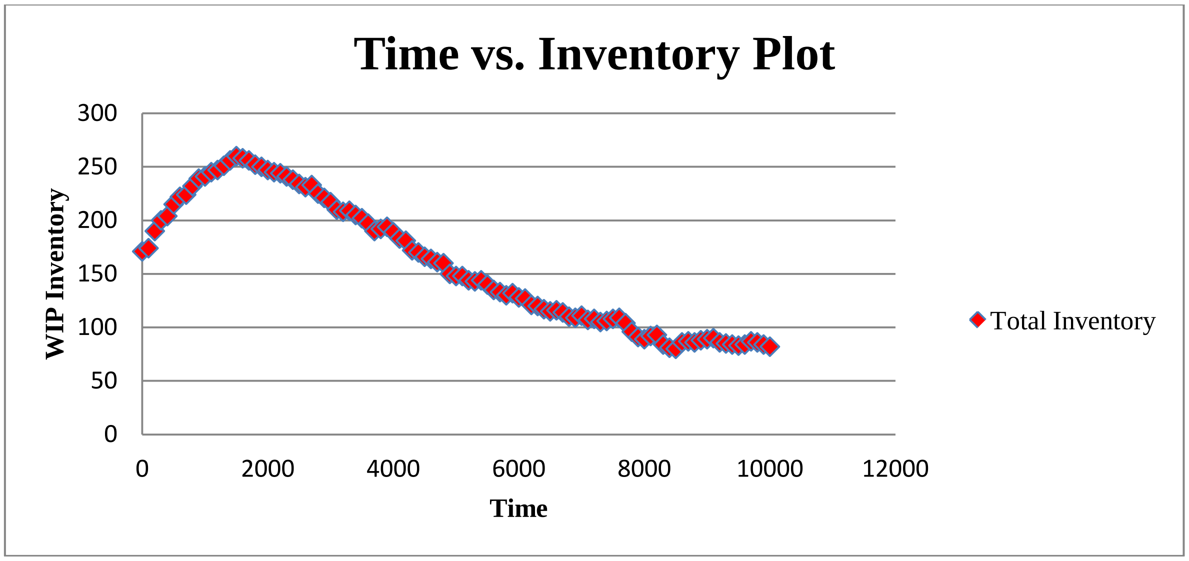

Figure 8: Time vs. WIP inventory

5. Conclusion and Future Research

In this paper we propose a RL algorithm to address inventory management problem in a disassembly line. The proposed algorithm is designed to be applied to a disassembly line with multiple products and demand arrivals. The aforementioned problem is highly dynamic where products and demands arrive on an uncertain pattern which continuously changes the state of the system. The highly adaptive Q-learning algorithm is the best fit for handling such dynamic and uncertain systems. However, defining the state space as the number of items (components, sub-assemblies and products) in the system and the number of demands for various components in the system leads to an explosion in the size of the state space. This challenge is handled by utilizing neural networks to estimate the Q-values of the state-action pairs encountered during the execution. We also obtained promising results when we compared the average inventory outputs obtained from RL and traditional push system.

Our next step in this research is to further analyzing the results obtained from the implementation of the proposed RL methodology. Our goal is to understand how agent behaves during the overall span of the execution to study the efficiency of RL. Also we intend to benchmark the RL methodology to a proven methodology selected from earlier studies in order to verify the efficiency of the proposed solution methodology. The proposed solution methodology is designed in a way that it can be applied to problems of similar nature such as demand driven disassembly systems and disassembly buffer allocation problem.

References

Bakker B. Reinforcement Learning by Backpropagation through an LSTM Model/Critic [Conference]. - [s.l.] : Proceedings of the IEEE symposium on Approximate Dynamic Programming and Reinforcement Learning, 2007. - pp. 127-134.

Johar B. and Gupta S. M. Balancing Inventory Generated from a Disassembly Line: Mathematical Approach [Conference]. - [s.l.] : Proceedings of SPIE, the International Society for Optical Engineering, 2006. - Vol. 6385. - pp. 638502.1-638502.9.

McGovern S. and Gupta S. M. The Disassembly Line - Balancing and Modeling [Book]. - New York City: McGraw-Hill, 2011

Qiao J., Hou Z. and Ruan X. Q-learning Based on Neural Network in Learning Action Selection of Mobile Robot [Conference]. - Jian : Proceedings of the IEEE International Conference on Automation and Logistics, 2007. - pp. 263-267.

Russell Stuart and Norvig Peter Artificial Intelligence: A Modern Approach [Book]. - Eaglewood Cliffs : Prentice Hall, 1995.

Shiraga N., Ozawa S. and Abe S. A Reinforcement LEarning Algorithm for Neural Networks with Incremental LEarning Ability [Conference]. - [s.l.] : Proceedings of the International Conference on Neural Information Processing, 2002. - Vol. 5. - pp. 2566-2570.

Shiraga N., Ozawa S. and Abe S. LEarning Action-Value Functions using Neural Networks with Incremental LEarning Ability [Conference]. - [s.l.] : Proceedings of the Annual Conference of the Institute Systems, Control and Information Engineers, 2001. - Vol. 45. - pp. 11-12.

Sun W., Wang X. and Cheng Y. Reinforceemnt Learning Method for Continuous State Space Based on Dynamic Neural Network [Conference]. - [s.l.] : Proceedings of the 7th World Congress on Intelligent Control and Automation, 2008. - pp. 750-754.

Sutton Richard S. and Barto Andrew G. Reinforcement Learning: An Introduction [Book]. - Cambridge : The MIT Press, 1998.

Tewari Ambuj Reinforcement Learning in Large or Unknown MDPs // Ph.D. Dissertation. - [s.l.] : University of California, Berkeley, 2007.

Udomsawat G. and Gupta S. M. Multi Kanban Mechanism for Appliance Disassembly [Conference]. - [s.l.] : Proceeding of the SPIE International Conference on Envrionmentally Conscious Manufacturing V, 2005. - pp. 30-41.

Udomsawat G. and Gupta S. M. Multi Kanban Model for Dsiassembly Line with Demand Fluctuation [Conference]. - [s.l.] : Proceedings of the SPIE International Conference on Environmentally Conscious Manufacturing III, 2003. - pp. 85-93.

Watkins C. J. C. H. Learning from Delayed Rewards // Ph.D. thesis. - [s.l.] : Cambridge University, 1989.

Watkins Charles and Dayan Peter Q-learning [Journal] // Machine Learning. - 1992. - pp. 279-292.