



Scholars' Mine

Masters Theses


Student Theses and Dissertations

Summer 2017

A new reinforcement learning algorithm with fixed exploration for semi-Markov decision processes

Angelo Michael Encapera

Follow this and additional works at: https://scholarsmine.mst.edu/masters_theses

 Part of the [Artificial Intelligence and Robotics Commons](#), and the [Operations Research, Systems Engineering and Industrial Engineering Commons](#)

Department:

Recommended Citation

Encapera, Angelo Michael, "A new reinforcement learning algorithm with fixed exploration for semi-Markov decision processes" (2017). *Masters Theses*. 7736.

https://scholarsmine.mst.edu/masters_theses/7736

This thesis is brought to you by Scholars' Mine, a service of the Missouri S&T Library and Learning Resources. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

A NEW REINFORCEMENT LEARNING ALGORITHM
WITH FIXED EXPLORATION FOR SEMI-MARKOV
DECISION PROCESSES

by

ANGELO MICHAEL ENCAPERA

A THESIS

Presented to the Faculty of the Graduate School of the
MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

2017

Approved by

Dr. Abhijit Gosavi, Advisor
Dr. David Enke
Dr. Zeyi Sun

© 2017
Angelo Encapera
All Rights Reserved

ABSTRACT

Artificial intelligence or machine learning techniques are currently being widely applied for solving problems within the field of data analytics. This work presents and demonstrates the use of a new machine learning algorithm for solving semi-Markov decision processes (SMDPs). SMDPs are encountered in the domain of Reinforcement Learning to solve control problems in discrete-event systems. The new algorithm developed here is called iSMART, an acronym for *imaging Semi-Markov Average Reward Technique*. The algorithm uses a constant exploration rate, unlike its precursor R-SMART, which required exploration decay. The major difference between R-SMART and iSMART is that the latter uses, in addition to the regular iterates of R-SMART, a set of so-called imaging iterates, which form an image of the regular iterates and allow iSMART to avoid exploration decay. The new algorithm is tested extensively on small-scale SMDPs and on large-scale problems from the domain of Total Productive Maintenance (TPM). The algorithm shows encouraging performance on all the cases studied.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Abhijit Gosavi, for his wonderful guidance, encouragement and support throughout my research and graduate studies. He has helped me develop the skills and analytical thinking needed to undertake graduate research.

I am also grateful to Dr. Zeyi Sun and Dr. David Enke for serving on my committee in spite of their busy schedules. I would also like to acknowledge the support of faculty members and staff of the Department of Engineering Management and Systems Engineering at the Missouri University of Science and Technology, who have, in one way or another, contributed to the completion of this thesis.

Lastly, I would like to thank my loving family and all who have played a supportive role in my life during my graduate studies. I could not have completed this thesis without the support and encouragement they have given me.

TABLE OF CONTENTS

	Page
ABSTRACT	iii
ACKNOWLEDGMENTS	iv
LIST OF ILLUSTRATIONS	vii
LIST OF TABLES	viii
NOMENCLATURE	ix
 SECTION	
1. INTRODUCTION	1
1.1. MDPS, SMDPS, DP, AND RL.....	1
1.2. ISMART	3
2. BACKGROUND MATERIAL AND A LITERATURE REVIEW	5
2.1. REINFORCEMENT LEARNING	5
2.2. TPM	9
3. TPM AND PRODUCTION INVENTORY SYSTEMS	12
4. iSMART ALGORITHM	17
4.1. Q-LEARNING	18
4.2. iSMART ALGORITHM.....	20
5. NUMERICAL RESULTS	23
5.1. SMALL-SCALE SMDP SYSTEMS	23
5.1.1. System 1.	24
5.1.2. System 2.	24
5.1.3. System 3.	24

5.1.4. System 4.	24
5.1.5. Numerical Results.	25
5.2. SMALL-SCALE PM RESULTS.....	25
5.3 LARGE-SCALE PM RESULTS.....	35
6. CONCLUSIONS AND FUTURE WORK.....	37
BIBLIOGRAPHY.....	38
VITA.....	41

LIST OF ILLUSTRATIONS

	Page
Figure 2.1: The Eight Pillars of TPM (Ahuja and Khamba, 2008).....	11
Figure 3.1: A Schematic Representation of a Single Production Machine System (from Gosavi et al, 2002).....	12
Figure 4.1: Simulation/Algorithm Schematic	21
Figure 5.1: Optimal Maintenance Policies Based on Production Cycles and Inventory Levels Example	29
Figure 5.2: R-Smart and iSMART's Performance Compared to DP-Optimal Solutions CS ₁	33
Figure 5.3: R-Smart and iSMART's Performance Compared to DP-Optimal Solutions on CS ₂	34
Figure 5.4: R-Smart and iSMART's Performance Compared to Small-Scale MTBF DP Optimal Solutions CS ₃	35
Figure 5.5: R-SMART and iSMART's Performance Compared to DP Optimal Solutions	36

LIST OF TABLES

	Page
Table 5.1: Results from Small-scale SMDPs.....	25
Table 5.2: Small-Scale PM Input Parameters	27
Table 5.3: Large-Scale PM Input Parameters	27
Table 5.4: Cost and Profit Parameters	28
Table 5.5: Results with Small-Scale System using CS_1	31
Table 5.6: Results on Small-Scale Systems using CS_2	32
Table 5.7: Results on Small-Scale Systems using CS_3	32
Table 5.8: Results on Large-Scale Systems for all Cost Structures.....	36

NOMENCLATURE

Symbol	Description
MDP	Markov Decision Process
SMDP	Semi-Markov Decision Process
DP	Dynamic Programming
RL	Reinforcement Learning
PM	Preventative Maintenance
TPM	Total Productive Maintenance
S	State space associated with decision process
n	Number of states
i, j	Indices for state in the state space S
$A(i)$	Action space for state i
a, b	An action in the action space
L	Lower limit of the buffer
U	Upper limit of the buffer
k	Number of iterations
α, β	Step sizes
Cm	Maintenance cost
Cr	Repair cost
CS_i	The i th maintenance, repair, and profit cost structure
ρ	Average reward
ρ^*	Optimal average reward

$p(i, a, j)$	The probability of going from state i to j under action a
$r(i, a, j)$	The reward for going from state i to j under action a
$t(i, a, j)$	The time spent in going from state i to j under action a
$Q(\dots)$	Q -Factor
$R(\dots)$	R-Factor
$T(\dots)$	T-Factor

1. INTRODUCTION

1.1. MDPS, SMDPS, DP, AND RL

Markov decision processes (MDPs) are problems of sequential decision-making in discrete-event systems controlled by the so-called Markov chains. In particular, MDPs seek to solve problems of optimal control. Using a controller to specify the action selected in a given state of the system, one can optimize system performance via consideration of quantifiable performance metrics, e.g., maximizing the net rewards obtained or minimizing the net costs incurred from operating the system. Typically, these performance metrics are defined in two ways: the average reward, where there is no discounting of money with time, and the discounted reward, where the time value of money is taken into account, i.e. discounting of money is considered. MDPs can be observed in many real-world applications, however, they are limited by the assumption that the so-called transition times in the problem should be constant.

Semi-Markov decision processes (SMDPs) are more generalized versions of MDPs. Unlike MDPs, SMDPs take time into consideration, i.e. the time of transition does not have to be constant. Hence, in SMDPs, the time spent in each state is treated as a random variable. In SMDPs, the time of transitions is also modeled within the objective function. In this thesis, the focus is on infinite time-horizon problems, where one assumes that the system will be observed for a very long time and will eventually settle into a steady state. Usually, the performance metrics used to study SMDPs are the same as those for MDPs: average reward and discounted reward. However, in MDPs time is not taken into

consideration in formulating the objective functions; rather, the objective functions are formulated in terms of transitions of the underlying Markov chains.

Both MDPs and SMDPs can be solved using dynamic programming (DP) when the number of state-action pairs is small enough, e.g., up to about 200. The two most popular methods of DP are: value iteration (Bellman, 1957) and policy iteration (Howard, 1960). However, “when the number of state-action pairs is too large,” transition probabilities cannot be generated (Ghosh, 2013) or stored in a computer; then DP is no longer a viable option for solving these problems. When this is the case, Reinforcement Learning (RL) techniques can be used.

RL is a relatively new simulation-based method for solving MDPs and SMDPs underlying the statistical model of the system. An advantage of these RL-based models is that they do *not* require the transition probabilities that must be estimated in the traditional DP approach. RL bypasses the tedious process of estimating the transition probabilities, but instead needs a discrete-event simulator to generate near-optimal solutions. Commercial software such as ARENA and MATLAB can be used to write these programs. In this thesis, the case study on which a new RL algorithm is tested is drawn from the domain of Total Productive Maintenance (TPM).

This thesis focusses on the presentation of a new simulation-based Reinforcement Learning (Bertsekas and Tsitsiklis, 1997; Sutton and Bartow, 1998; Gosavi, 2014a) algorithm for solving SMDPs (Puterman, 2005; Bertsekas, 2000). In particular, the algorithm developed here is tested on a preventive maintenance problem encountered in production-inventory (PI) systems (Das and Sarkar, 2000).

Total Productive Maintenance (TPM) was first developed in Japan in the 1970s (Ghosh, 2013). Its goal is to deliver higher utilization of machines. This improves the availability of production machines, and the frequency of unexpected failures decreases. Unexpected failures both increase lead times and the overall operating cost. By using TPM, efficient preventive maintenance (PM) schedules can be generated, thus decreasing the frequency of unexpected failures, without compromising on the volume of production before maintenance is performed. Proper implementation of TPM can lead to the saving of millions of dollars over the years (McKone and Weiss, 1997).

1.2. ISMART

The SMDP under consideration here employs the so-called average reward problem in which one seeks to maximize the net profits earned per unit time over an infinitely long time horizon. An existing RL algorithm for solving SMDPs under this average reward criterion is called R-SMART (Gosavi, 2004), which is known to require decay of the so-called exploration parameter (or exploration rate). Unless this parameter is decayed carefully during the runtime of the algorithm, the latter usually fails to generate the optimal or near-optimal solution. Hence, developed this thesis is a new version of the R-SMART algorithm, called iSMART, which does not require the decay of this exploration parameter, but instead works with a fixed rate of exploration.

The remainder of the thesis is organized as follows: 2 briefly reviews the literature on TPM, production inventory systems, and reinforcement learning techniques, as well as provides a background of the research conducted in this work to the reader. Section 3

describes in more detail the production inventory systems used in this work. Section 4 in detail the reinforcement learning techniques and specifically the iSMART algorithm proposed in this work. Section 5 details the numerical results obtained from running the iSMART algorithm in the PI simulator. The final Section of this thesis presents closing remarks for the work and proposes possible avenues for future work.

2. BACKGROUND MATERIAL AND A LITERATURE REVIEW

In order to explore the research issues surrounding the iSMART algorithm and how it uses TPM in production inventory (PI) systems, a brief review of TPM, PI systems, MDPs, SMDPs, DP, iSMART's predecessor, and R-SMART, along with a review of the relevant literature, is presented here.

2.1. REINFORCMENT LEARNING

As stated in the first section, problems of sequential decision-making in discrete-event systems driven by Markov chains can often be modeled by MDPs and SMDPs. MDPs and SMDPs are control-optimization problems whose goal is to select the best action in each state visited by the system such that a pre-defined "performance metric is optimized for the discrete-event system driven by Markov chains" (Gosavi, 2014b). In such settings, the system jumps from one state to another, usually in a random manner, and the transitions follow the Markovian property (Gosavi, 2014b). In the MDP, when a system visits a state, a decision-maker or agent must select an optimal action from the set of multiple actions allowed in that state (Ghosh, 2013). For every state-to-state transition there exists a transition probability (TP). These TPs constitute an integral part of any MDP model; further they are "dependent on the state and action chosen in the state" (Gosavi, 2014b). An important feature of the MDP models is that the "probability of transitioning from one state to another" does not depend on the number or nature of transitions that have already taken place in the system (Gosavi, 2014b).

MDPs find numerous applications in operations management, e.g., queuing control (Sennott, 1999), supply chain management (Buffett, 2004), maintenance management (Schouten and Vanneste, 1995), vehicle routing (Su et al. 2011), revenue management (Lautenbacher and Jr, 1999) etc.; see Ghosh (2013) for additional examples. MDPs can be used in other areas of operations management too, e.g., finance (Feinberg and Shwartz, 2002), search algorithms (Amin et al. 2012), and robotic control (Abbeel et al. 2007). However, these applications are limited by discrete, equal time periods between events. When time is incorporated into the model, as a random variable, the SMDP is a more appropriate model than the MDP.

SMDPs are more generalized versions of MDPs. As stated before, in SMDPs the time spent in each state transition is a random variable. Because of this property, the MDP becomes a special case of the SMDP; an MDP is thus an SMDP where all state transition times are equal. Further, SMDPs use time as an element of the performance metric. Performance metrics under which SMDPs are studied include the so-called discounted reward and the expected reward under a finite or infinite time horizon. In this work, expected reward under an infinite time horizon, also called average reward, will be studied. Like MDPs, SMDPs have many real-world applications, including queueing control, maintenance management, vehicle routing, etc.

In the setting considered in this thesis, when the system is in a given state, the decision-maker chooses an action. After a finite amount of time elapses, the system transitions to a new state where the decision-maker then selects a new action for the current state in which the system finds itself. These transitions usually have a cost or a reward associated with them. However, under the SMDP property, the probability of transition

from state to state relies only on the current state and the action being chosen in that state; increasing the reward or cost associated with these decisions does not affect the probability of transitions. This important property is key to using DP (Gosavi, 2014b). It should be noted that extensions of DP techniques meant for MDPs are also useful in solving SMDPs. The two well-known DP methods include value iteration and policy iteration. Value iteration, which is more popular because it is easier to code and understand, is studied in this work.

“DP, developed by Bellman (1957) and Howard (1957), is a field that provides algorithms” to solve MDPs/SMDPs (Gosavi, 2014b). This work led to the creation of the famous Bellman equation for optimality. Related equations developed by Howard (1957) are called the Poisson equation or the Bellman equation *for a given policy* (Gosavi, 2014b). DP methods are effective on problems in which the best decisions can be found sequentially. Using the Bellman optimality equation, a number of optimization problems useful for solving many real-world problems can be constructed. To apply these DP models, one needs

- The set of possible states visited by the system
- The set of possible actions allowed in each state
- TPs for each action
- The transition reward function for each action
- The transition time function for each action

DP is a very useful and effective tool. However, it can be limited by the size of the system. It can be effective in systems with relatively few states, e.g. up to 100. However, it begins to break down beyond a few states. A system with 1000 states would yield a TP matrix that

contains 1000×1000 elements. Computers are generally not capable of storing matrices exceeding a million elements. Also, creating TPs for real world systems can quickly become very tedious. Further, creating the TPs for these systems is often not a straightforward process, especially for systems with numerous input random variables. Solving MDPs/SMDPs *without* generating the TPs for the system is clearly hence a desirable goal, thereby providing motivation for RL. (Gosavi, 2014b).

RL, as stated earlier, is a simulation based technique that seeks to solve “MDPs/SMDPs when TPs are not available” (Ghosh, 2013) or are not obtainable in practice. The usage of RL techniques allows the study of the effect of actions on the system. These simulated results yield net cumulative rewards earned during a state-transition.

RL has gained a significant amount of popularity in the artificial intelligence and machine learning communities. It has been able to find optimal or near-optimal solutions for systems that DP cannot be applied. As stated above, R-SMART, an existing RL algorithm for solving average reward SMDPs, is known to converge to near-optimal solutions for large-scale systems. However, it requires a so-called exploration rate that needs to decay with time. This tuning parameter, i.e., the exploration rate, makes the algorithm less than ideal in terms of practical applications. The iSMART algorithm proposed here seeks to alleviate this difficulty by eliminating this tuning parameter altogether from the algorithm, instead using a constant exploration rate. The iSMART algorithm will be discussed in much greater detail later in the thesis.

2.2. TPM

TPM originated from the fields of reliability and maintenance. These closely related fields have given birth to many standard engineering practices in numerous industries (McKone and Weiss, 1998). TPM takes a structured look at production systems in order to make scheduled maintenance a necessary part of the standard practices in production systems. TPM's goal is to improve utilization of production resources (Ghosh, 2013). Thus, one major desirable end result of using TPM is to reduce the "frequency of repairs or unexpected failures of machines" (Ghosh, 2013); failures worsen lead times for production cycles, eventually increases the total operating costs (McKone and Weiss, 1998).

As stated previously, TPM was first developed by the Japanese manufacturing industry in the 1970s. It was introduced to the United States in the late 1980s for a variety of reasons.

TPM is usually implemented in multiple phases. Many cost-saving decisions can be made in the first two phases by taking different approaches to machine maintenance. Ultimately, these cost saving decisions focus on the idea of decreasing the mean and variance of the production life cycle time. The lifecycle times can be reduced using a variety of strategies proposed in the literature, including "autonomous maintenance investment decisions to reduce inventory" (McKone and Weiss, 1997) and "one-time investments to improve process quality and reduce set up time" (Porteus, 1986). Also important to reducing life cycle costs is determining when to undertake maintenance. This decision-making problem can be modeled by Markov decision processes (Marcellus and Dada, 1994). McCall (1965) discusses two main maintenance models; policies for systems where the failure distributions are known and for systems where the failure distributions are unknown.

When the distributions are known the models can be broken down into two subsets. The first scenario is one where the system fails stochastically and the actual state is unknown. This means that maintenance can only happen when repair or inspection is scheduled, and uncertainty is due to the inability to predict the exact time of failure. The second sub-set presented in McCall (1965) has stochastic system failure, however, the actual state is known. This allows for immediate reaction to system failure. The uncertainty in this scenario comes from the inability to predict the exact time of failure. When the failure distributions are not known, there are many ways to handle the uncertainty. Jorgenson and McCall (1963) discusses methods for a variety of such scenarios.

Basic practices of TPM are often referred to as elements or pillars of TPM. TPM is most effective when all of its eight pillars are present to support the practice (Sangameshwaran and Ranganathan, 2002). By implementing these suggested pillars, as recommended by the Japan Institute of Plant Maintenance (JIPM), the following effects are often observed: increase in labor productivity, and reduced maintenance costs, production stoppages and downtimes (Ahuja and Khamba, 2008). These core TPM initiatives classified into the so-called eight pillars are as follows: Autonomous Maintenance; Focused Maintenance; Planned Maintenance; Quality Maintenance; Education and Training; Office TPM; Development Management; and Safety, Health and Environment HSE (Ireland and Dale, 2001; Shamsuddin et al., 2005; Rodrigues and Hatakeyama, 2006). This eight-pillar approach is depicted in Figure 2.1 which also provides a visual representation of the central idea.

As stated before, TPM has been found to be a very cost-effective tool for manufacturing operations with the ability to save firms millions of dollars. However,

finding optimal solutions to maintenance problems with very large state spaces can be a challenging problem. Reinforcement Learning methods can offer innovative and effective ways to solve these large-scale problems.

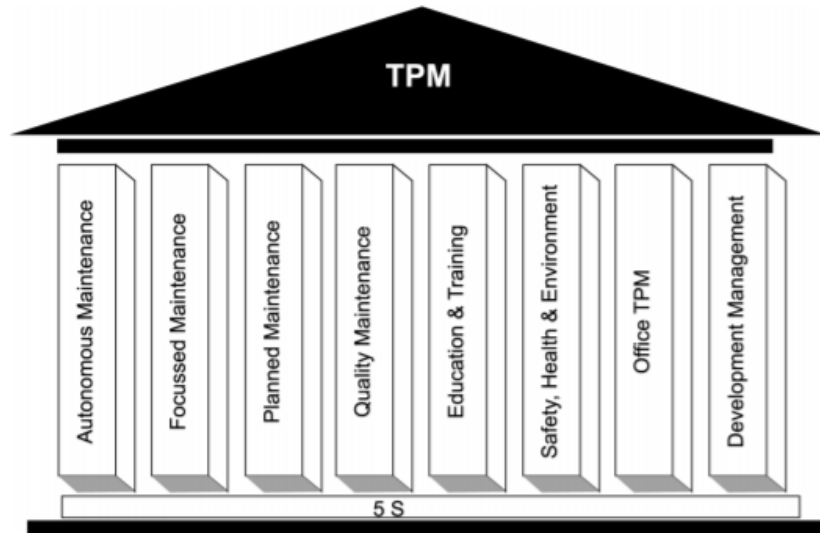


Figure 2.1: The Eight Pillars of TPM (Ahuja and Khamba, 2008)

3. TPM AND PRODUCTION INVENTORY SYSTEMS

In this section, details of the TPM case study related to the PI system are provided. Consider the make-to-stock, single machine (Askin and Goldberg, 2002), PI system shown in Figure 3.1. This PI system produces a single product unit with the goal of meeting the external demand. Since it is assumed that the system can fail, TPM methods can be used to decrease the cost of operation (Das and Sarkar, 1999). Also, the time between failures, which is a random variable, is *not* exponentially distributed; the distribution used in this case study is the gamma distribution. This makes it necessary to employ preventative maintenance (Lewis 1994) as a vehicle to reduce the downtime of the machine.

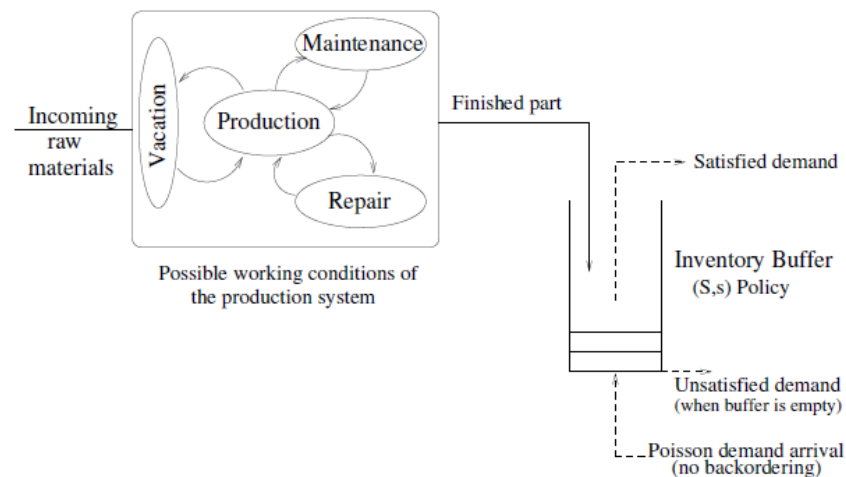


Figure 3.1: A Schematic Representation of a Single Production Machine System (from Gosavi et al, 2002)

The goal of the SMDP model is to optimize the maintenance schedule for the PI system in such a way that minimizes the net average cost of running the system. Indirectly,

by minimizing the net average cost of running the system, the frequency of machine failures is minimized while the number of production cycles completed before preventive maintenance needs to be performed is maximized. In these types of production systems, when a manager decides that PM needs to take place, the system must be shut down for maintenance. The time interval after which the system must be shut down for maintenance varies from machine to machine and usually acquires a unique value for each machine depending on its failure characteristics. When a machine is down for maintenance, inventory cannot be produced. When the system is not creating inventory, it may become impossible to meet demands. This is why in addition to actual monetary costs, maintenance has an unmeasured opportunity cost associated with it. Thus, maintenance must be carefully planned so that machines are not over-maintained.

The so-called production cycle associated with the machine ends after one unit of the product has been manufactured. With every consecutive completed production cycle that has occurred without any maintenance or repairs, the probability of system failure generally increases. In this work, the number of consecutive production cycles completed without maintenance is used as a performance metric. This will be discussed in more detail in the results section.

As stated previously, a major advantage of using RL methods to solve PI system problems is the ability to use simulation-based techniques, which can be used on complex real-world problems for which analytical models do not exist. To generate the PI system simulator, a series of distributions are needed for a variety of different inputs to the model including: the production times, the repair times, the maintenance times, the time between

demands, and the time between failures. Each of these random variables is customizable and allows for a variety of different distributions to be tested relatively easily.

The production times, the time between failures, and the repair times for this model are assumed to belong to the Erlang distribution. The Erlang distribution is very customizable. This distribution allows the user to alter both the shape and scales in such a way that allows the model to be very flexible. Having this flexibility is key in creating a robust simulation. The Erlang distribution is a two-parameter family of continuous probability distributions where the random variable's value is always greater than or equal to 0. The two parameters used in this distribution are the "shapes" (k) and the "rates" (λ). Often in place of the 'rate', the so-called scale, denoted by $(1/\lambda)$, is used. The scale is simply the inverse of the rate.

The time between demands is modeled using the exponential distribution. The exponential distribution can be viewed as a special case of the Erlang distribution in which the shape parameter (k) is equal to one and the rate is altered to change the values of the distribution.

The Production-Inventory system is of the make-to-stock kind where the unit produced after completion of a production cycle is placed in the (finished product) inventory buffer. When a demand arrives, the demand is satisfied, and the inventory buffer is depleted by one if there the buffer is not empty. However, if there is no inventory in the buffer, the demand cannot be satisfied and the opportunity to sell a product is lost. In this case study, the assumption is that the demand does not wait until there is inventory to satisfy the demand, it simply leaves unsatisfied. There are upper and lower limits associated with the inventory buffer; this implies that when the buffer reaches its upper limit,

production is stopped and is not started again until the buffer reaches its lower limit. When the upper limit is reached, the system goes on “vacation”. This means that the system does not continue to create inventory, and is on down time until enough demand has arrived to satisfy the condition defined by the lower inventory buffer limit.

A fixed profit is associated with each demand satisfied. Fixed costs are also assigned to maintenance and repair costs. The profit for a demand satisfied and costs associated with maintenance and repair are also customizable parameters of the PI system simulation. This allows for multiple cost benefit ratios to be tested. It is also assumed that after the system is maintained or repaired, it delivers the same performance as a totally new machine or system.

The state of the PI system is defined by (φ, ω) , where φ denotes the number of consecutive production cycles completed without repair or maintenance and ω denotes the number of units in the buffer. After every successful production cycle, a decision must be made to either produce a product or maintain the machine. Thus, there are two actions that can be taken: either (1) produce or (2) maintain. Production can only be chosen when the inventory buffer is below the upper limit (U); otherwise, the system will “go on vacation.” The system will stay on vacation until demands arrive and lower the amount of inventory in the buffer to its lowest limit (L) at which the production must start.

The different possible transitions are described next. The system can either progress from $(\varphi = i)$ to $(\varphi = i + 1)$ by selecting the action of production and successfully completing the production cycle, or it can progress from $(\varphi = i)$ to $(\varphi = 0)$ if the machine

fails during the production cycle. When the action maintain is chosen, the system will go from $(\varphi = i)$ to state $(\varphi = 0)$. When a failure occurs, the system also goes from its current state to $\varphi = 0$.

4. iSMART ALGORITHM

In this section, the iSMART algorithm is described in detail. A different version of this algorithm appeared in Ghosh (2013). This thesis proposes a new version of iSMART that does not employ the contraction factor originally found in Ghosh’s version of iSMART (Ghosh, 2013). Although iSMART is a variant of R-SMART (Gosavi, 2004), the new algorithm proposed here is expected to behave in a more robust manner in comparison to R-SMART. The reason for this is that “the decaying of the so-called exploration rate” needed for R-SMART is not needed here; R-SMART’s behavior depends on how well a tuning parameter, which determines the exploration rate, is gradually reduced (Ghosh, 2013). Full exploration essentially allows the algorithm to select every action in each state with the same probability. Fixed exploration implies that the probability of selecting an action is not changed. Thus, full exploration implies fixed exploration but not vice-versa. In simulators, it is often easy to run the algorithm with full or fixed exploration.

When a tuning parameter of this nature that controls the exploration is introduced into a RL algorithm, the algorithm can no longer be considered “fully exploratory”. Further, the decaying of the exploration itself typically requires a rule. R-SMART’s behavior depends on selecting the right rule for exploration, i.e., the right tuning of this exploration rate (parameter). This makes any algorithm with such a tuning parameter less robust in terms of its behavior. In fact, if the tuning is not done properly, R-SMART even fails to converge to optimal solutions on small problems. Thus, a major contribution of this work is to present a new variant of R-SMART that performs with fixed exploration

and still manages to converge to optimal solution (problems with small state-spaces) or near-optimal solutions (problems with large state spaces).

4.1. *Q*-LEARNING

The iSMART algorithm is considered to be a *Q*-learning algorithm since it is based on value iteration, which is a dynamic programming technique. Value iteration, when used to solve average reward SMDPs (and MDPs also), requires a so-called “uniformizing” technique that requires transition probabilities (TPs). However, when these TPs are not known, RL can be applied because RL works in simulators and does not require the transition probabilities. RL algorithms based on value iteration require the so-called *Q*-factors, and, in addition to these *Q*-factors, iSMART needs a dual *image* of the main *Q*-factors, which are stored in two separate sets of iterates. These dual images will be called the R- and T-factors in this thesis. This *image* (i) is the inspiration behind the suffix in the name iSMART.

As stated before, like any other *Q*-learning algorithm, iSMART is based on value iteration. This allows the Bellman optimality equation to be the underlying foundation for determining the optimal solution using *Q*-learning algorithms. In other words, the solutions generated by iSMART are expected to reach those of the Bellman optimality equation, which is known to generate the optimal solution (Puterman, 1994). In this thesis the *Q*-factor version of the Bellman optimality equation (Bellman, 1957) is needed, as follows in Equation (4.1).

$$Q(i, a) = \sum_{j=1}^{|S|} p(i, a, j) \left[r(i, a, j) - \rho^* t(i, a, j) + \max_{b \in \mathcal{A}} Q(j, b) \right] \quad (4-1)$$

for all $i \in S$ and $a \in \mathcal{A}(i)$.

Using the above equation, a value iteration update can be derived, which is as follows:

$$Q(i, a) \leftarrow \sum_{j=1}^{|S|} p(i, a, j) \left[r(i, a, j) - \rho^* t(i, a, j) + \max_{b \in \mathcal{A}(i)} Q(j, b) \right] \quad (4-2)$$

for all $i \in S$ and $a \in \mathcal{A}(i)$.

However, it should be noted that the above equation is difficult to use in practice because ρ^* is not known from the start. In order to resolve this issue, the following equation can be used. For all $i \in S$ and $a \in \mathcal{A}(i)$,

$$Q(i, a) \leftarrow \sum_{j=1}^{|S|} p(i, a, j) \left[r(i, a, j) - \tilde{\rho} t(i, a, j) + \max_{b \in \mathcal{A}(i)} Q(j, b) \right]. \quad (4-3)$$

In the above equation, $\tilde{\rho}$ denotes an estimate of ρ^* where $\tilde{\rho}$ is slowly updated and should eventually converge to ρ^* . Note that this term, ρ , is an estimate of the current average reward. Due to this, iSMART will not only update Q -factors, but it will also need to update values of ρ . The update of will take place using the so-called “mirror image” concept discussed above.

The mirror image will constitute of R and T factors that will essentially pursue the greedy action stored in the Q -factors, i.e., the first set of iterates. A *separate* set of R and T factors will be used to update on a second time scale, which will use a *different step size* and *follow the greedy policy* from the first set of iterates. Using this mirror image ensures that if (i^*, a^*) denotes a distinguished state-action pair frequently visited in the simulator,

then $R(i^*, a^*)/T(i^*, a^*)$ should converge to ρ^* in the limit, i.e., as the number of iterations converges to infinity. Note that the following description of the algorithm follows that of Ghosh (2013), with a notable difference that the contracting factor η used there is eliminated here. Eliminating the contracting factor, i.e., setting $\eta = 1$, makes a big difference to the computational performance of the algorithm, which was never tested in Ghosh (2013) on large-scale problems. The iSMART algorithm was tested on only small-scale problems in Ghosh (2013).

4.2. iSMART ALGORITHM

A step-by-step explanation of the iSMART algorithm will now be presented. A schematic of the process is provided in Figure 4.1.

Step 1: Set the number of iterations, k , to 1. Set $\rho^k = 0$, where ρ^k is the estimation of the optimal average reward in the k^{th} iteration. Set $Q^k(i, a)$, $R^k(i, a)$, and $T^k(i, a)$ to 0 for all $i \in S$ and all $a \in \mathcal{A}(i)$. Set k_{max} to a large number that will allow the algorithm to successfully explore all states and actions. Set (i^*, a^*) to any state-action pair in $S \times \mathcal{A}$ (preferably a state-action pair that is visited *frequently such as* $(0, 1)$.)

Step 2: Start the system simulation at an arbitrary state i . Select an action with a probability of $\frac{1}{|\mathcal{A}(i)|}$. Note that this probability is the exploration rate that was discussed before. This probability is never changed during the course of the algorithm, but may have to be set to a value other than $\frac{1}{|\mathcal{A}(i)|}$, depending on the nature of the problem. This will be discussed in more detail later in the work.

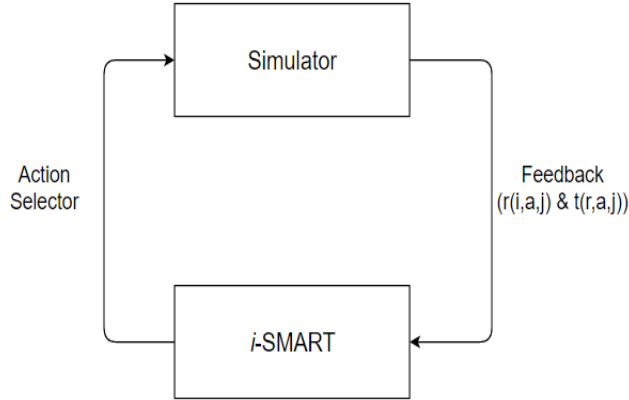


Figure 4.1: Simulation/Algorithm Schematic

Step 3: Simulate action a . Let the next state be denoted by j . Let $r(i, a, j)$ denote the immediate reward from state i to state j under action a . Also let $t(i, a, j)$ denote the time spent under the same state-action transition.

Step 4: Update the Q -factor via Equation (4.4). The term α is a step size that should be chosen suitably, and must remain positive. In this work, multiple step sizes were tested. Only the most effective step size was reported.

$$Q^{k+1}(i, a) \leftarrow [1 - \alpha^k]Q^k(i, a) + \alpha^k \left[r(i, a, j) - \rho^k t(i, a, j) + \max_{b \in \mathcal{A}(j)} Q(j, b) \right] \quad (4.4)$$

Step 5: Compute μ^{k+1} , where $\mu^{k+1} = \arg \max_{a \in \mathcal{A}(i)} Q^{k+1}(i, a)$ for all $i \in S$.

Step 6: If $a \in \arg \max_{a \in \mathcal{A}(i)} Q^{k+1}(i, a)$ (i.e., the action selection was a greedy one), update $R^k(i, a)$ and $T^k(i, a)$ as follows via Equations (4.5) and (4.6), respectively. The term β is a step size that should be chosen suitably, and must remain in the interval (0,1).

$$R^{k+1}(i, a) \leftarrow [1 - \beta^k]R^k(i, a) + \beta^k \left[t(i, a, j) - R^k(i^*, a^*) + \max_{b \in \mathcal{A}(j)} R^k(j, b) \right] \quad (4.5)$$

$$T^{k+1}(i, a) \leftarrow [1 - \beta^k]T^k(i, a) + \beta^k \left[t(i, a, j) - T^k(i^*, a^*) + \max_{b \in \mathcal{A}(j)} T^k(j, b) \right] \quad (4.6)$$

Step 7: Update ρ^{k+1} using Equation (4.7) as follows:

$$\rho^{k+1} = R^{k+1}(i^*, a^*)/T^{k+1}(i^*, a^*). \quad (4.7)$$

Step 8: If $k < k_{max}$, set $i \leftarrow j$ and $k \leftarrow k+1$ and return to Step 2. Otherwise, continue to Step 9.

Step 9: For each $l \in S$, compute $d(l) \in \arg \max_{a \in \mathcal{A}(l)} Q^k(l, a)$. The policy returned by the algorithm is d , where action in state l is given by $d(l)$.

5. NUMERICAL RESULTS

In this section, the numerical results produced by iSMART are detailed for (i) a small-scale SMDP with only four state-action pairs and (ii) a PM case study discussed in the TPM section with a maximum of approximately 30 million state-action pairs. For the SMDP small-scale systems, four cases were studied. For the PM case study, thirteen cases were investigated. These thirteen cases are classified into “small” time-between-failure cases and “large” time-between-failure cases. Data for these thirteen cases that were studied were obtained from the literature (Das and Sarkar, 1999). A subset of these results were presented in Encapera and Gosavi (2017).

The rest of this section is organized as follows. Section 5.1 contains details of the small-scale SMDP. Section 5.2 describes the numerical results obtained with the small time-between-failure cases for the TPM case study, while Section 5.3 describes the same for the large time-between failure cases.

5.1. SMALL-SCALE SMDP SYSTEMS

The input data for each of the four small-scale case studied is provided in the Subsections 5.1.1 through 5.1.4. In each of these subsections, P_a , TRM_a , and TTM_a denote the transition probability matrix, transition reward matrix, and transition time matrix for action a , respectively. Note that $P_a(i, j) = p(i, a, j)$, where $P_a(i, j)$ denotes the element in the i^{th} row and j^{th} column of the matrix P_a . Similarly, $TRM_a(i, j) = r(i, a, j)$, where

$TRM_a(i, j)$ denotes the element in the i^{th} row and j^{th} column of the matrix TRM_a ; and $TTM_a(i, j) = t(i, a, j)$, where $TTM_a(i, j)$ denotes the element in the i^{th} row and j^{th} column of the matrix TTM_a .

5.1.1. System 1.

$$P_1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \quad P_2 = \begin{bmatrix} .9 & .1 \\ .2 & .8 \end{bmatrix}$$

$$TRM_1 = \begin{bmatrix} 6 & -5 \\ 7 & 12 \end{bmatrix} \quad TRM_2 = \begin{bmatrix} 10 & 17 \\ -14 & 13 \end{bmatrix}$$

$$TTM_1 = \begin{bmatrix} 10 & 5 \\ 120 & 60 \end{bmatrix} \quad TTM_2 = \begin{bmatrix} 50 & 75 \\ 7 & 2 \end{bmatrix}$$

5.1.2. System 2.

$$P_1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \quad P_2 = \begin{bmatrix} .9 & .1 \\ .2 & .8 \end{bmatrix}$$

$$TRM_1 = \begin{bmatrix} 6 & 5 \\ 7 & 12 \end{bmatrix} \quad TRM_2 = \begin{bmatrix} 10 & 17 \\ 14 & 13 \end{bmatrix}$$

$$TTM_1 = \begin{bmatrix} 10 & 5 \\ 120 & 60 \end{bmatrix} \quad TTM_2 = \begin{bmatrix} 5 & 75 \\ 7 & 20 \end{bmatrix}$$

5.1.3. System 3.

$$P_1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \quad P_2 = \begin{bmatrix} .9 & .1 \\ .2 & .8 \end{bmatrix}$$

$$TRM_1 = \begin{bmatrix} 6 & -5 \\ 70 & 12 \end{bmatrix} \quad TRM_2 = \begin{bmatrix} 12 & 17 \\ 6 & 13 \end{bmatrix}$$

$$TTM_1 = \begin{bmatrix} 10 & 5 \\ 120 & 60 \end{bmatrix} \quad TTM_2 = \begin{bmatrix} 50 & 75 \\ 7 & 20 \end{bmatrix}$$

5.1.4. System 4.

$$P_1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \quad P_2 = \begin{bmatrix} .9 & .1 \\ .2 & .8 \end{bmatrix}$$

$$TRM_1 = \begin{bmatrix} 16 & 5 \\ 75 & 120 \end{bmatrix} \quad TRM_2 = \begin{bmatrix} 80 & 10 \\ 6 & 1 \end{bmatrix}$$

$$TTM_1 = \begin{bmatrix} 10 & 5 \\ 120 & 60 \end{bmatrix} \quad TTM_2 = \begin{bmatrix} 50 & 75 \\ 7 & 20 \end{bmatrix}$$

5.1.5. Numerical Results. The results of using the new algorithm on simulators written in MATLAB and run on a 2.60 GHz Intel Core i7-6700HQ processor on a 64-bit Windows operating system. Table 5.1 highlights the optimal policies for each SMDP system. These benchmark optimal policies were determined using policy iteration. Table 5.1 includes the optimal policies using policy iteration iSMART was able to identify the correct optimal policy for all four small systems in the simulators.

Table 5.1: Results from Small-scale SMDPs

SMDP system	Optimal Policy
1	(1,2)
2	(2,2)
3	(1,1)
4	(2,1)

5.2. SMALL-SCALE PM RESULTS

The time between demand (TBD) arrivals follows the exponential distribution in this thesis, following the original source of data (Das and Sarkar, 1999). This is typical of a lot of work in supply chain management, where it has been found that demand size over long intervals of time follows the normal distribution. The normal distribution can be approximated by the Poisson distribution for the number of arrivals, which implies that the

time between demand arrivals has to be exponentially distributed. Here μ will denote the mean rate of arrival; the mean for the exponential distributed inter-arrival time will be $1/\mu$. It has been shown that when the system experiences increasing failure rates, i.e., as time progresses the probability of failure increases, then gamma distributions are good models (Lewis, 1994). Due to this fact, the Erlang distribution is often used to model time between failures. In the case study used (Das and Sarkar, 1999), the Erlang distribution is used for the time between failures and also for the time of production and the repair time. The Erlang distribution is one whose *pdf* resembles that of a hill and is of a general nature that can accommodate many distributions that have a double tapering nature. The mean of the Erlang distribution (n, λ) is given by n/λ , and the variance is given by n/λ^2 . By looking at the mean time between failures, the classifications for “small” PM cases and “large” PM cases can be made. The small cases as seen in Table 5.2 have mean time between failures (MTBF) 100 time units or less. If the MTBF is greater than 100, it is considered a large case. Here, the maintenance time, or the time it will take to complete maintenance activities, will be assumed to follow a uniform distribution (a,b) . The inventory limit is defined by the upper and lower bounds (L, U) . These inputs for the different small and large case systems studied are specified in Table 5.2 and Table 5.3 respectively. The cost values used for the simulations include the cost of maintenance, cost of repair, and profit from a unit sale. Although all values in the following table are positive, the simulation treats costs as negative profits. There are three different cost structures used to study the effectiveness of the iSMART algorithm. These three cost structures are defined in Table 5.4.

The cost values used for the simulations include the cost of maintenance, cost of repair, and profit from a unit sale. Although all values in the following table are positive, the simulation treats costs as negative profits. There are three different cost structures used to study the effectiveness of the iSMART algorithm. These three cost structures are defined in Table 5.4.

Table 5.2: Small-Scale PM Input Parameters

System	Time Bet. Demands ($1/\mu$)	Time Bet. Failure (n,λ)	Time for Production (n,λ)	Maint. Time (a,b)	Repair Time (n,λ)	Inventory Limits (L,U)
1	10	(8,0.08)	(8,0.8)	(5,20)	(2,0.01)	(2,3)
2	5	(8,0.08)	(8,0.8)	(5,20)	(2,0.01)	(2,3)
3	7	(8,0.08)	(8,0.8)	(5,20)	(2,0.01)	(2,3)
4	15	(8,0.08)	(8,0.8)	(5,20)	(2,0.01)	(2,3)
5	20	(8,0.08)	(8,0.8)	(5,20)	(2,0.01)	(2,3)
6	10	(4,0.1)	(8,0.8)	(5,20)	(2,0.01)	(2,3)
7	10	(4,0.08)	(8,0.8)	(5,20)	(2,0.01)	(2,3)
8	10	(8,0.08)	(4,0.4)	(5,20)	(2,0.01)	(2,3)
9	10	(8,0.08)	(8,0.8)	(2,10)	(2,0.01)	(2,3)
10	10	(8,0.08)	(8,0.8)	(5,20)	(1,0.05)	(2,3)

Table 5.3: Large-Scale PM Input Parameters

System	Time Bet. Demands ($1/\mu$)	Time Bet. Failure (n,λ)	Time for Production (n,λ)	Maint. Time (a,b)	Repair Time (n,λ)	Inventory Limits (L,U)
11	10	(4,0.01)	(8,0.8)	(5,20)	(2,0.01)	(2,3)
12	10	(8,0.008)	(8,0.8)	(5,20)	(2,0.01)	(2,3)
13	10	(8,0.08)	(4,0.8)	(5,20)	(2,0.01)	(2,3)

Table 5.4: Cost and Profit Parameters

Cost Structure (CS)	CS 1	CS 2	CS 3
Maintanance Cost (Cm)	2	2	1
Repari Cost (Cr)	5	10	10
Unit Profit	1	0.5	100

The results obtained from the iSMART algorithm are provided in the form of an optimal policy and the average reward. The optimal policy for each system is denoted by 3 integers: (i_1, i_2, i_3) . This notation implies that if the inventory level is c , the optimal action is to produce until the production count is less than i_c and to maintain when the production count equals i_c for $c = 1, 2$, and 3. This can be better explained via Figure 5.1 where the y -axis is the production count at which maintenance should be performed, and the x -axis is the inventory level. In this example, the optimal solution would be recorded as (8,6,4). This means that when the inventory level is 1, the system should be maintained after 8 production cycles; if the inventory level is 2, the system should be maintained after 6 production cycles; when the inventory level is 3, the system should be maintained after 4 production cycles. The optimal policy description given here and displayed in this work omits inventory levels of 0 because when the inventory level is 0, the optimal action will always be to produce (Das and Sarkar, 1999).

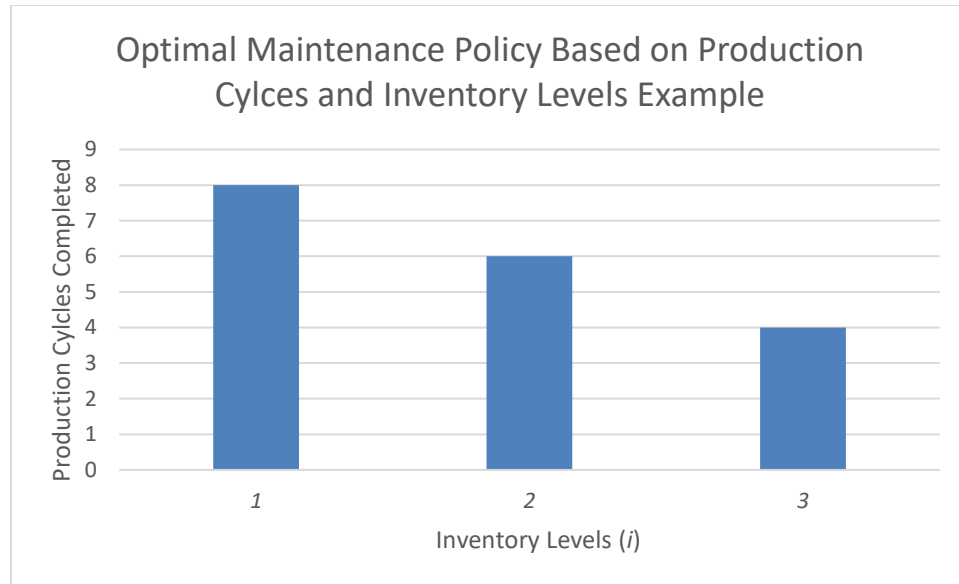


Figure 5.1: Optimal Maintenance Policies Based on Production Cycles and Inventory Levels Example

The so-called exploration rate used in the experimentation is also provided in the tables related to the results. The exploration rate is the probability that the production action will be selected by the action selector during the simulation. Also critical for the algorithm's success are the so-called step sizes, represented by α and β . The same step sizes are used for both large and small systems, as well as in all three cost structures. These step-sizes, α and β , are given by Equations (5-1) and (5-2) respectively.

$$\alpha = 150/(300+k) \quad (5-1)$$

$$\beta = 10/(300+(3*k)) \quad (5-2)$$

In the above equations, k is the current time point in the simulation. Due to this, these step sizes approach zero as the simulation progresses.

The optimal policy can be found by inspecting the Q -values (see Step 8 in the iSMART algorithm in Section 4 for additional details). The algorithm was implemented in the simulator for a duration of 1 million time units for each system; this portion of the simulation is called the learning simulation phase. The optimal policy was computed from inspecting the Q -values found at the end the of learning simulation phase. The simulator was then run again using the optimal policy for 100,000 time units with 8 replications. This is called the frozen phase. The learning and frozen phases typically took 45 and 27 seconds, respectively, on a 2.60 GHz Intel Core i7-6700HQ processor on a 64-bit Windows operating system.

The upper and lower bounds for 95% confidence intervals (CI) on the mean average reward (ρ) are also provided along the mean in the tables that depict the result. The optimal policies displayed for each system were obtained using DP (Das and Sarkar, 1999). In the small-scale PM systems using cost structure one, the values of ρ were obtained using DP (Das and Sarkar, 1999) However, the optimal values of ρ for all other systems and cost structures are found using the optimal policies in the frozen phases. This helps eliminate simulation error when comparing optimal values

The exploration rates *theoretically* should be at 0.5 when there are two actions. However, in our experiments, for most of our systems, a higher probability (0.65) was used for the production action to ensure that the algorithm thoroughly explored the state-action space. In other words, during the simulation trials, these systems should be producing more frequently than they are being repaired or being maintained. In the large-scale PM systems, as defined earlier, where the mean time between failure (MTBF) is larger than that of the others (greater than 100 time units), the exploration rate needed to be adjusted to an even

greater probability (0.95). It can be concluded that systems with large MTBF values have optimal policies that are in states with large production counts, and without the bias towards production, the algorithm would never explore these states, which would cause the algorithm to converge to sub-optimal policies.

To benchmark the effectiveness of iSMART, the R-SMART algorithm was tested using a standard decay of the exploration probability of $10/(100 + k)$, with a starting exploration probability of 0.99. This was done to demonstrate R-SMART's dependency on a decaying exploration rate. However, fine-tuning the exploration can get R-SMART to deliver optimal performance. As stated earlier, the advantage of iSMART is that it does *not* require the fine tuning or decay of the exploration rate. Tables: 5.5:7 display the simulation results using iSMART and R-SMART for all small-scale cases and large-scale cases.

Table 5.5: Results with Small-Scale System using CS₁

System	Optimal ρ^*	Optimal Policy	iSmart Exploration Rate	iSmart Policy	<i>iSmart Mean</i> (ρ)	R-Smart Policy	R-Smart Exploration rate	<i>R-Smart Mean</i> (ρ)
1	0.0296	(6,5,5)	0.65	(6,5,6)	0.0296 ± 0.0015	(5,2,6)	0.99	0.0286 ± 0.0006
2	0.0237	(6,6,5)	0.65	(6,5,4)	0.0234 ± 0.0009	(4,4,4)	0.99	0.0244 ± 0.0006
3	0.0273	(6,5,5)	0.65	(5,5,6)	0.0283 ± 0.0009	(3,5,5)	0.99	0.0261 ± 0.0007
4	0.0267	(6,6,5)	0.65	(5,5,5)	0.0284 ± 0.0005	(4,2,6)	0.99	0.0205 ± 0.0003
5	0.0232	(6,6,6)	0.65	(6,6,4)	0.0232 ± 0.0007	(3,4,6)	0.99	0.0164 ± 0.0004
6	-0.0054	(4,4,4)	0.65	(3,3,6)	0 ± 0.0006	(3,2,3)	0.99	-0.0056 ± 0.0004
7	-0.00011	(4,4,4)	0.65	(4,4,5)	-0.0001 ± 0.0007	(4,2,2)	0.99	-0.0005 ± 0.0005
8	0.0287	(6,6,5)	0.65	(5,4,5)	0.0295 ± 0.0006	(4,2,5)	0.99	0.0263 ± 0.0007
9	0.0261	(7,6,5)	0.65	(5,5,8)	0.0266 ± 0.0011	(3,5,12)	0.99	0.0218 ± 0.0004
10	0.0413	(8,8,6)	0.65	(6,6,6)	0.0417 ± 0.0006	(4,5,250)	0.99	0.0253 ± 0.0007

Table 5.6: Results on Small-Scale Systems using CS₂

System	Optimal ρ^*	Optimal Policy	iSmart Exploration Rate	iSmart Policy	<i>iSmart Mean</i> (ρ)	R- Smart Policy	R-Smart Exploration rate	<i>R-Smart Mean</i> (ρ)
1	-0.0046 ± 0.001	(6,5,5)	0.65	(5,6,7)	-0.0035 ± 0.0004	(5,2,6)	0.99	-0.0066 ± 0.0004
2	-0.0108 ± 0.0005	(6,5,5)	0.65	(5,6,7)	-0.0105 ± 0.0008	(4,3,12)	0.99	-0.0105 ± 0.0008
3	-0.0076 ± 0.0007	(6,6,5)	0.65	(6,6,6)	-0.0076 ± 0.0007	(4,3,5)	0.99	-0.0071 ± 0.0008
4	-0.0022 ± 0.0007	(6,5,5)	0.65	(6,6,7)	-0.0015 ± 0.0007	(4,2,3)	0.99	-0.0093 ± 0.0005
5	-0.0002 ± 0.0003	(5,5,5)	0.65	(5,5,6)	-0.0002 ± 0.0003	(3,4,4)	0.99	-0.0069 ± 0.0002
6	-0.0335 ± 0.0010	(4,4,4)	0.65	(5,2,3)	-0.0338 ± 0.0006	(2,4,250)	0.99	-0.0345 ± 0.0009
7	-0.0290 ± 0.0005	(4,4,4)	0.65	(5,4,4)	-0.0290 ± 0.0005	(2,3,3)	0.99	-0.0310 ± 0.0012
8	-0.0047 ± 0.0007	(5,5,5)	0.65	(6,6,7)	-0.0044 ± 0.0008	(3,3,10)	0.99	-0.0086 ± 0.0005
9	-0.0034 ± 0.0006	(5,5,5)	0.65	(5,5,5)	-0.0034 ± 0.0006	(3,3,5)	0.99	-0.0061 ± 0.0004
10	-0.0055 ± 0.0013	(6,5,5)	0.65	(5,5,5)	-0.0055 ± 0.0007	(4,4,7)	0.99	-0.0059 ± 0.0009

Table 5.7: Results on Small-Scale Systems using CS₃

System	Optimal ρ^*	Optimal Policy	iSmart Exploration Rate	iSmart Policy	<i>iSmart Mean</i> (ρ)	R- Smart Policy	R-Smart Exploration rate	<i>R-Smart Mean</i> (ρ)
1	-0.1682 ± 0.0053	(5,5,5)	0.65	(5,5,6)	-0.1682 ± 0.0053	(2,3,100)	0.99	-0.194 ± 0.0039
2	-0.2418 ± 0.0045	(5,5,5)	0.65	(5,4,3)	-0.243 ± 0.0062	(3,3,2)	0.99	-0.2391 ± 0.0034
3	-0.2064 ± 0.0030	(5,5,5)	0.65	(5,5,6)	-0.2064 ± 0.0030	(3,4,250)	0.99	-0.2045 ± 0.0030
4	-0.1231 ± 0.0031	(5,5,5)	0.65	(5,5,6)	-0.1231 ± 0.0031	(4,3,6)	0.99	-0.1331 ± 0.0037
5	-0.0934 ± 0.0029	(5,5,5)	0.65	(4,5,7)	-0.0919 ± 0.0043	(2,3,4)	0.99	-0.1608 ± 0.0026
6	-0.3794 ± 0.0053	(3,1,1)	0.65	(2,2,2)	-0.3818 ± 0.0057	(1,2,250)	0.99	-0.3790 ± 0.0072
7	-0.3488 ± 0.0113	(3,2,2)	0.65	(2,1,3)	-0.3501 ± 0.0046	(1,2,4)	0.99	-0.3522 ± 0.0078
8	-0.1667 ± 0.0048	(5,5,5)	0.65	(4,4,4)	-0.1669 ± 0.0034	(3,3,10)	0.99	-0.1758 ± 0.0047
9	-0.1527 ± 0.0087	(5,5,5)	0.65	(5,5,3)	-0.1527 ± 0.0087	(2,3,3)	0.99	-0.1525 ± 0.0032
10	-0.2096 ± 0.0064	(4,4,4)	0.65	(4,4,4)	-0.2096 ± 0.0064	(4,4,4)	0.99	-0.2096 ± 0.0064

iSMART's performance under CS₁ was very strong. Figure 5.2 below shows the proportion of small-scale MTBF systems studied under CS₁ that either outperformed, matched, or performed worse than the optimal solutions found using DP for both iSMART and R-SMART-derived results. Under these conditions, iSMART outperformed 50% of

the small-scale MTBF systems, and statistically matched the optimal solutions on the remaining 50% using a 95% confidence interval. Under these conditions, R-SMART did not fare nearly as well. R-SMART only statistically outperformed the optimal solution on 10% of the systems, matched on 10% of the systems, and statistically performed worse on the remaining 80% of the systems using CS₁.

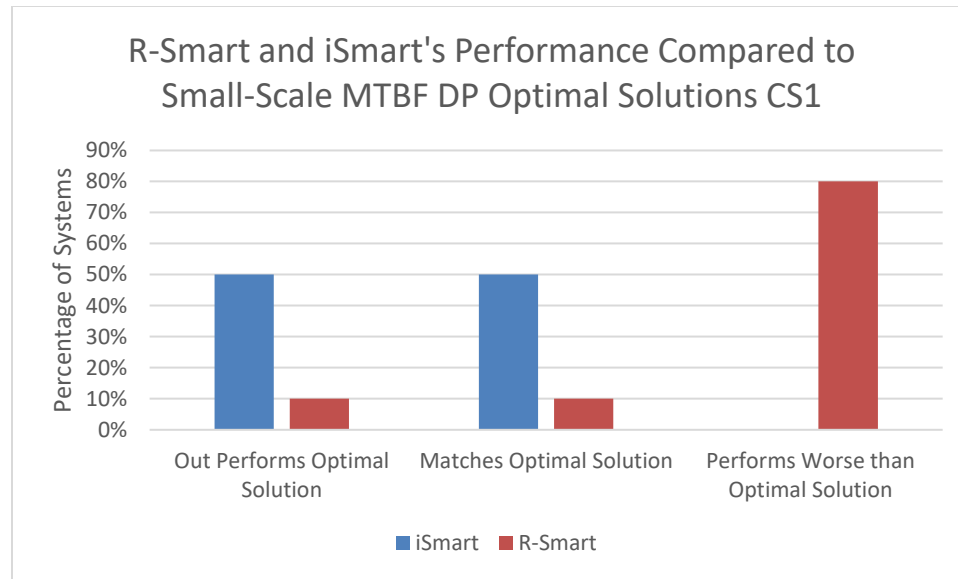


Figure 5.2: R-Smart and iSMART's Performance Compared to DP-Optimal Solutions CS₁

Under CS₂, iSMART again performed very well. Figure 5.3 below shows the proportion of small-scale MTBF systems studied under CS₂ that either outperformed, matched, or performed worse than the optimal solutions found using DP for both iSMART as well as R-SMART-derived results. Under these conditions, iSMART outperformed 10% of the small-scale MTBF systems, and statistically matched the optimal solutions on

the remaining 90% using a 95% confidence interval. R-SMART did not outperform any optimal solutions, and statistically matched on only 40% of these systems. R-SMART performed statistically worse on the remaining 60% of these systems.

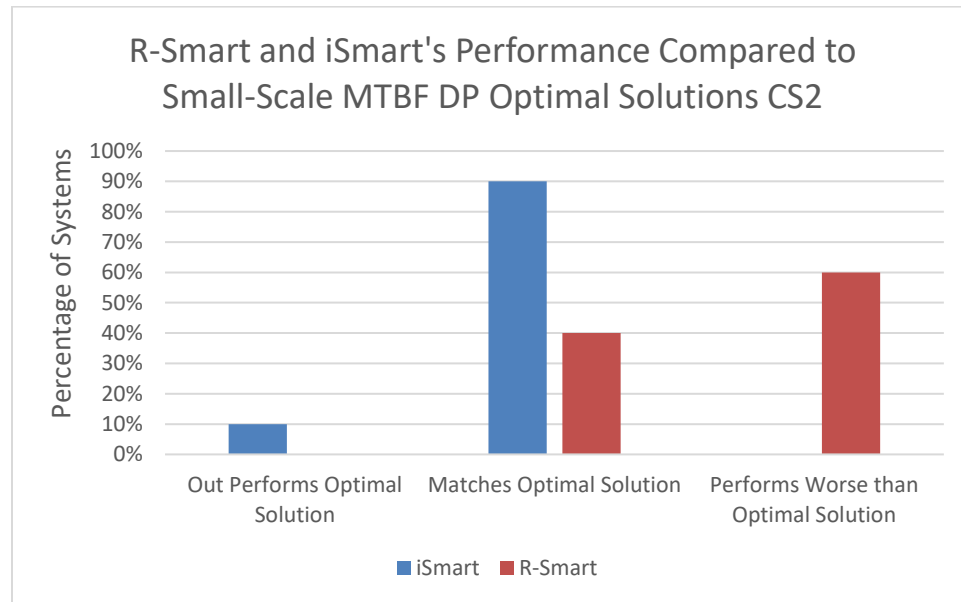


Figure 5.3: R-Smart and iSMART's Performance Compared to DP-Optimal Solutions on CS₂

Under CS₃, iSMART performed adequately. Figure 5.4 below shows the proportion of small-scale MTBF systems studied under CS₃ that either outperformed, matched, or performed worse than the optimal solutions found using DP for both iSMART as well as R-SMART derived results. Under these conditions, iSMART statistically matched 100% of the optimal solutions using a 95% confidence interval. iSMART likely did not outperform any of the optimal solutions because of the little room for improvement CS₃ leaves. In this cost structure, the cost of repair is 100 times greater than the profit earned

from filling a demand. R-SMART did not outperform any optimal solutions, and statistically matched on only 60% of these systems. R-SMART performed statistically worse on the remaining 40% of these systems.

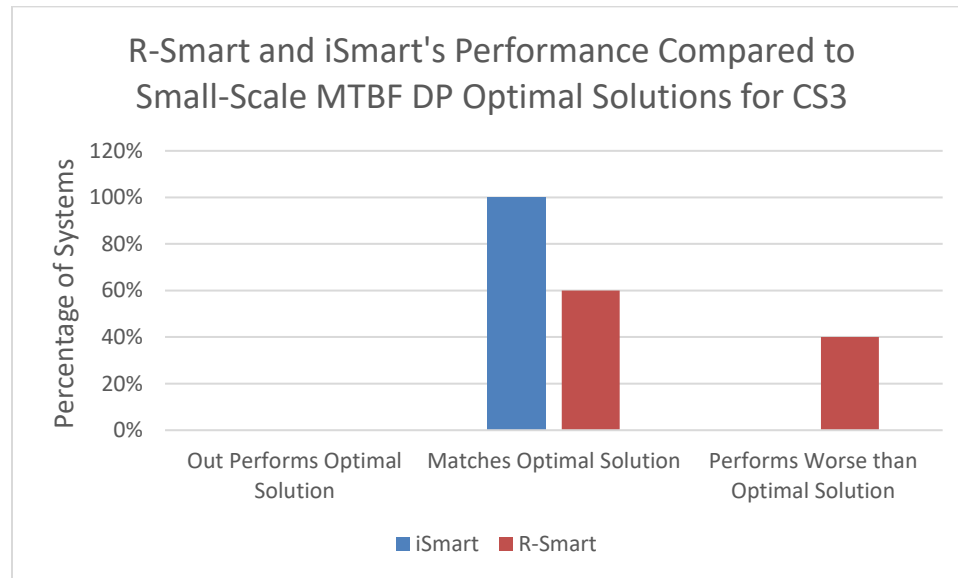


Figure 5.4: R-Smart and iSMART's Performance Compared to Small-Scale MTBF DP Optimal Solutions CS₃

5.3 LARGE-SCALE PM RESULTS

The full results for the large-scale MTBF systems can be seen in Table 5.8. iSMART performed well in these systems. Figure 5.5 below shows the percentage of large-scale MTBF systems studied under CS₁, CS₂, and CS₃ that either outperformed, matched, or performed worse than the optimal solutions found using DP for both the iSMART and R-SMART derived results. Under these conditions, iSMART out performed 11% of the large-scale MTBF systems, and statistically matched the optimal solutions on the

remaining 89% using a 95% confidence interval. R-SMART did not outperform any optimal solutions, and statistically matched on only 56% of these systems. R-SMART performed statistically worse on the remaining 44% of these systems.

Table 5.8: Results on Large-Scale Systems for all Cost Structures

System	Optimal ρ^*	Optimal Policy	iSmart Exploration Rate	iSmart Policy	<i>iSmart Mean</i> (ρ)	R-Smart Policy	R-Smart Exploration rate	<i>R-Smart Mean</i> (ρ)
CS1-11	0.0616 ± 0.0016	(21,19,13)	0.95	(21,19,8)	0.0616 ± 0.0016	(42,40,39)	0.99	0.0577 ± 0.0005
CS1-12	0.0754 ± 0.0008	(63,59,41)	0.95	(58,57,69)	0.0758 ± 0.0009	(51,42,53)	0.99	0.0763 ± 0.0005
CS1-13	0.0655 ± 0.0006	(11,10,9)	0.95	(12,10,7)	0.0655 ± 0.0013	(9,5,7)	0.99	0.0643 ± 0.0008
CS2-11	0.0235 ± 0.0004	(19,18,15)	0.95	(19,20,22)	0.0235 ± 0.0011	(5,9,14)	0.99	0.0119 ± 0.0003
CS2-12	0.0354 ± 0.0006	(57,54,42)	0.95	(45,47,40)	0.0356 ± 0.0004	(60,57,46)	0.99	0.0356 ± 0.0004
CS2-13	0.0205 ± 0.0003	(11,10,10)	0.95	(10,10,8)	0.0205 ± 0.0008	(6,6,9)	0.99	0.0149 ± 0.0003
CS3-11	-0.0094 ± 0.0042	(15,14,14)	0.95	(14,14,19)	-0.0094 ± 0.0038	(5,15,22)	0.99	0.0119 ± 0.0003
CS3-12	0.0565 ± 0.0018	(47,45,35)	0.95	(38,61,52)	0.0570 ± 0.0011	(73,45,54)	0.99	0.0358 ± 0.0004
CS3-13	-0.0762 ± 0.0047	(27,9,9)	0.95	(6,6,8)	-0.0638 ± 0.0018	(6,6,7)	0.99	0.0149 ± 0.0003

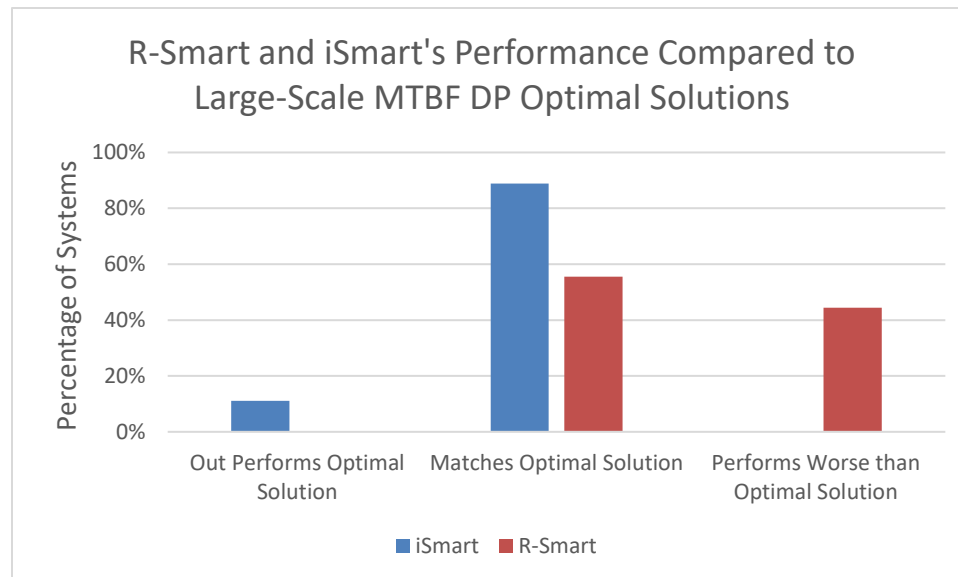


Figure 5.5: R-SMART and iSMART's Performance Compared to DP Optimal Solutions

6. CONCLUSIONS AND FUTURE WORK

This thesis proposed a new version of a RL algorithm known as R-SMART. The new version is called iSMART, and is based on a Q -factor version of the Bellman optimality equation (Bellman, 1957) that also uses a mirror imaging principle and is designed to overcome a critical deficiency of R-SMART, i.e., the need for decaying exploration. The proposed algorithm works with a fixed exploration rate. In particular, the SMART family of algorithms has been used gainfully on problems from the domain of TPM. TPM is known to save firms millions of dollars over the years for production firms. Without a TPM program, most production firms suffer from excessive downtimes, which can result in missing deadlines and increased costs of repairs. Therefore, iSMART was also implemented for solving TPM problems in production-inventory systems.

In the experiments conducted, iSMART was able to generate solutions that were statistically more profitable than the “optimal” solutions obtained from DP for 18% of all systems tested, and was statistically equivalent to optimal solutions obtained from DP for the other 82%.

Scope for future work: A natural extension of this work would be to increase inventory limits on the systems studied to test the algorithm’s ability to handle systems with even larger state-action spaces. In future work, mathematical convergence of the iSMART algorithm should also be studied. Further, studying non-Poisson arrivals and multiple machines will lead to other exciting avenues for future research.

BIBLIOGRAPHY

Abbeel, P., Coates, A., Quigley, M., and Ng, A. Y. (2007). An application of reinforcement learning to aerobatic helicopter flight. In Proceedings of Advances in Neural Information Processing Systems 19. MIT Press.

Ahuja, I.P.S. and Khamba, J.S. (2008), Total Productive Maintenance: Literature Review and Directions. International Journal of Quality & Reliability Management, Vol. 25 No. 7, pp. 709-56.

Amin, K., Kearns, M., Key, P., and Schwaighofer, A. (2012). Budget optimization for sponsored search: Censored learnings in MDPs. In Proceedings of the Twenty-eighth Conference on Uncertainty in Artificial Intelligence. UAI.

Bellman, R. (1957). Dynamic Programming. Princeton University Press, New Jersey, USA.

Bertsekas, D. (2000). Dynamic Programming and Optimal Control. Athena, Belmont, 2nd edition

Bertsekas, D., and Tsitsiklis, J. (1997). Neuro-Dynamic Programming. Athena Press, Cambridge, MA, USA.

Buffett, S. and Scott, N. (2004). An algorithm for procurement in supply-chain management. In Proceedings of the AAMAS 2004 Workshop on Trading Agent Design and Analysis.

Das, T. and Sarkar, S. (1999). Optimal preventive maintenance in a production inventory system. IIE Transactions, 31:537–551.

Encapera, A. and Gosavi, A. A new Reinforcement Learning algorithm with fixed exploration for semi-Markov control in preventive maintenance. Proceedings of the ASME 2017 12th International Manufacturing Science and Engineering Conference, MSEC2017, June 4-8, 2017, Los Angeles, CA, USA.

Feinberg, E. and Shwartz, A. (2002). Handbook of Markov Decision Processes: Methods and Application. Kluwer, Massachusetts, USA.

Ghosh, S. (2013). Two Essays on Dynamic Programming and Reinforcement Learning (Doctoral dissertation). Doctoral Dissertations. Paper 2431.

http://scholarsmine.mst.edu/doctoral_dissertations/2431

- Gosavi, A. (2004) Reinforcement learning for long-run average cost. *European Journal of Operational Research*, 155, 654-674.
- Gosavi, A. (2014a) *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*, Second Edition, Springer, New York, NY, USA.
- Gosavi, A. (2014b) How to Rein in the Volatile Actor: A New Bounded Perspective , *Procedia Computer Science*, Complex Adaptive Systems Conference, Philadelphia, PA, Volume 36, pp. 500-507.
- Howard, R. (1960). *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, Massachusetts, USA.
- Ireland, F. and Dale, B.G. (2001), A study of total productive maintenance implementation. *Journal of Quality in Maintenance Engineering*, Vol. 7 No. 3, pp. 183-91.
- Jorgenson, D. W. and J. J. McCall (1963), Optimal Scheduling of Replacement and Inspection. *Operations Research*, 11, 723-747.
- Lautenbacher, C. and Jr, S. S. (1999). The underlying Markov decision process in the single-leg airline yield-management problem. *Transportation Science*, 33:136–146.
- Marcellius, R. L. and M. Dada (1991), Interactive Process Quality Improvement. *Management Science*, 37, 11, 1365-1376.
- McCall, J. J. (1965) Maintenance Policies for Stochastically Failing Equipment: A Survey. *Management Science*, 11, 5, 493-524.
- McKone, K. E., and Weiss, E. (1998) TPM: Planned and Autonomous Maintenance: Bridging the Gap Between Practice and Research, *Production and Operations Management*, 7:4, 335-351.
- Porteus, E. L. (1986), Optimal Lot Sizing, Process Quality Improvement and Setup Cost Reduction, *Operations Research*, 34, 1, 137-144.
- Puterman, M. (2005). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, New York, USA.
- Rodrigues, M. and Hatakeyama, K. (2006), Analysis of the fall of TPM in companies. *Journal of Materials Processing Technology*, Vol. 179 Nos 1-3, pp. 276-9.

Sangameshwaran, P. and Jagannathan, R. (2002), HLL's manufacturing renaissance. *Indian Management*, November, pp. 30-5.

Schouten, F. V. D. D. and Vanneste, S. (1995). Maintenance optimization of a production system with buffer capacity. *European Journal of Operational Research*, 82(2):323–338.

Sennott, L. (1999). *Stochastic Dynamic Programming and the Control of Queueing Systems*. John Wiley and Sons, New York, USA.

Shamsuddin, A., Hassan, M.H. and Taha, Z. (2005), TPM can go beyond maintenance: excerpt from a case implementation. *Journal of Quality in Maintenance Engineering*, Vol. 11 No. 1, pp. 19-42.

Su, Z., Jiang, J., Liang, C., and Zhang, G. (2011). Path selection in disaster response management based on Q-learning. *International Journal of Automation and Computing*, 8(1):100–106.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction* MIT Press, Cambridge, MA, USA.

VITA

Angelo Encapera was born on May 27, 1994 in Wichita, Kansas. He received his B.S in Petroleum Engineering from the Missouri University of Science and Technology, Rolla, Missouri in May 2016. Immediately after completing his undergraduate studies, Mr. Encapera began his graduate studies in the Fall of 2016 at the Missouri University of Science and Technology.

Mr. Encapera has submitted and presented a conference paper at the MSEC 2017 conference at the University of Southern California. His research interests include reinforcement learning, total productive maintenance, and data sciences. Angelo is a member of INFORMS and has presented at the INFORMS annual meeting in 2016 and 2017. Mr. Encapera received his M.S in Systems Engineering from Missouri University of Science and Technology in December of 2017.