Applications of the Learning Automata Algorithm of Thathachar and Sastry

Aug, 31, 2005
Technical Report SOPTL-05-02
Missouri University of Science and Technology

Abhijit Gosavi GOSAVIA@MST.EDU
219 Engineering Management
Department of Engineering Management and Systems Engineering
Missouri University of Science and Technology
Rolla, MO 65409, USA

Tolga Tezcan
University of Illinois at Urbana-Champaign

Sakkapak Wichinrojjarun

Abstract

In recent times, a number of machine-learning and pattern-recognition methods have been used to solve operations-management problems in the real world. In this report, we focus on a powerful *learning automata* algorithm, from the field of pattern-recognition and machine vision, to solve stochastic discrete-optimization problems encountered in the industry. The algorithm is due to Thathachar and Sastry [51]. We present an empirical study of this technique on two real-world test-beds of interest in engineering management. The first test-bed, which is from production management, is a complex buffer-optimization problem that defies exact theoretical analysis. The other test-bed is related to revenue management from the airline domain where also it is difficult to develop an exact theoretical model. We find that the learning automata technique produces encouraging computational results. We also discuss some analytical properties of the algorithm.

Keywords: learning automata, buffer optimization, revenue management

1. Introduction

Artificial-intelligence (AI) and information-technology (IT) paradigms have been used for data mining, forecasting, and manufacturing control etc for some time now. They are emerging as powerful players in solving *management science problems* as well. The advantages of using them on management-science problems are more obvious in complex domains where theoretical models are *not* easily obtainable. Many theoretical models break down in the face of realistic assumptions making them less useful in solving complex management-science problems arising in the real world. It is well-known that for stochastic systems, simulation-based models are capable of (i) accurately evaluating the function and (ii) accommodating real-world assumptions. In other words, where theoretical models break down,

simulation-based models can estimate the objective function accurately. On such problems, AI- and IT-based optimization paradigms become especially useful because they can be combined easily with simulation models. Combination of AI- and IT-based paradigms with simulation, hence, forms a powerful tool for solving complex real-world problems.

Models that do not exploit the closed form of the objective function but rely on numeric function evaluations are often referred to as *black-box* models. Oftentimes black-box models — especially those belonging to the AI family — are computationally intensive. But, because of the increasing power of computers and the ever-increasing need for generating more *accurate* models in the industry, the popularity of black-box paradigms is on the rise — especially in the field of IT. In recent times, machine learning and AI methodologies have made a dramatic impact in operations management. Examples of such methodologies, to name only a few, are: (i) Meta-heuristics and stochastic search (see [49] for a unified view), e.g., tabu search [18], simulated annealing [28] (SA), genetic algorithms [27], simultaneous perturbation [45], nested partions [42], ant colony [10], stochastic ruler [59], dyna-search [11], and GRASP [15], (ii) neural networks [25], e.g., neurons, backpropagation, and radial-basis networks, and (iii) reinforcement learning [48, 19], e.g., Q-Learning [57], SARSA [39, 47], and average-cost algorithms [23, 22, 20].

The theory of *learning automata* (LA), which is yet another AI-based methodology (see [33] for a comprehensive text; other important texts and lecture-notes series include [53, 34, 4, 29, 14]), is a powerful and untapped source of computational techniques that could be exploited for solving real-world problems in operations management. In this report, we have discussed how to adapt a largely-unknown, but powerful, pattern-recognition algorithm from the theory of LA to solve discrete optimization problems arising in industrial domains, e.g., production systems and airline systems. The algorithm we study here was introduced in Thathachar and Sastry [51], and unlike many LA algorithms, it can be used for multi-variate optimization. LA algorithms have stayed largely-restricted to IT-based domains, e.g., pattern-recognition and machine vision tasks [40, 16]. Some applications to management-science domains include [35, 54, 21]. In [36], a single-decision variable LA algorithm has been used for selecting the best system in simulations.

In this report, we first present an empirical testing of the multi-variate algorithm of [51] on two complex real-world problems — one from buffer management in transfer lines and the other from revenue management in the airline industry. Thereafter, we study some analytical properties of the algorithm.

The rest of this report is organized as follows. Algorithm details are provided in Section 2. Section 3 presents empirical results with the buffer-optimization problem while Section 4 presents results with the airline problem. Some analytical properties of the algorithm are studied in Section 5. Section 6 concludes the report with a discussion of the work presented here and directions for further research.

2. Algorithm Details

The LA algorithm that we focus on differs from and resembles other relevant algorithms in a number of ways. All LA algorithms belong to the family of stochastic search algorithms. But, as mentioned above, unlike a majority of other LA algorithms, the LA algorithm of [51]

can be used for *multiple* decision variables. Also, unlike most other meta-heuristics, LA is *not* a local search technique; as such, it does not need any neighborhood-generation scheme. LA's working mechanism in some sense is memory-based, which is similar to that of many other meta-heuristics. Also, like most meta-heuristics, its convergence is asymptotic, i.e., it converges to the optimal solution when the number of iterations tend to infinity. However, we will show later that the algorithm has a natural stopping criterion that can be exploited in practice.

Table 1: Relationship between AI and optimization terminologies

	LA	Optimization Theory
1.	Automaton	Decision Variable
2.	Policy or collection of actions	Solution
3.	Environment	Objective function
4.	Feedback (or response) from environment	Objective function value
5.	Learning rate	Step size

The theory of LA has its roots in adaptive behavioral psychology and learning. As such much of its terminology is heavily based on terms popular in IT and AI. Therefore, in Table 1, we provide a one-to-one map of its terminology with commonly-used terms in operations research (OR). We now provide an intuitive explanation of how LA works. In the data-base, it maintains a distribution of quantities which are updated in every iteration of the algorithm. The quantity in question will be denoted by $p^k(i,a)$ and will be used for selecting the value a for the ith decision variable in the kth iteration of the algorithm. We will show in Section 5 that this quantity, $p^k(i,a)$, is actually a probability measure. Further let I denote the number of decision variables, and let $\mathcal{A}(i)$ denote the set of values that the ith decision variable can assume. We will assume that $\mathcal{A}(i)$ is finite for any given i, but that $\prod_{i=1}^{I} |\mathcal{A}(i)|$, which is the size of the solution space, is too large for an exhaustive evaluation. In the beginning, this probability is assigned the same value for all parameters. That is, for any i, $p^1(i,a) = \frac{1}{|\mathcal{A}(i)|}$ for every a. Using these probabilities, a value is selected for each decision variable. These values together constitute a "solution." The algorithm updates these quantities based on the value of the objective function obtained from the solution. The updating mechanism is designed to ensure the following: *When a good solution is obtained, the probability of selecting it is increased, and when a poor solution is obtained, the probability of its selection is reduced.* Clearly, after each update, the algorithm gets smarter. As the algorithm progresses, it makes improved choices, and is gradually attracted to the optimal solution.

To be used in the updating mechanism, the objective function value has to be normalized to a number between 0 and 1 using the best and worst possible values for the objective function. The objective function value, G, is *normalized* as follows.

$$\frac{G - G_{\min}}{G_{\max} - G_{\min}},$$

where $G_{\max}$ and $G_{\min}$ denote the maximum (best) and the minimum (worst) objective function value, respectively, that can be obtained from the problem. The value of $G_{\max}$ and $G_{\min}$ may be available from performance-analysis bounds on the problem or else have to be "guesstimated," by the user.

In the algorithm description below, $B(i,a)$ will denote the best normalized objective function value — obtained so far in the algorithm— associated with selecting the value a for the ith decision variable. F_* will denote the best *normalized* objective function value found *so far* in the algorithm. The step size used in updating the quantities will be denoted by μ. The algorithm is written in terms of maximizing the value of the objective function.

2.1 Steps in the algorithm

1. Set the number of iterations, k, to 1. Also, set

$$p^1(i,a) = \frac{1}{|\mathcal{A}(i)|}, \text{ and } B(i,a) = 0$$

for $i = 1, 2 \ldots, I$ and every $a \in \mathcal{A}(i)$. Assign any positive value less than 1 to μ. $G_{\max}$ and $G_{\min}$ are to be assigned as discussed above. F_*, which denotes the best (normalized) objective function value obtained so far by the algorithm, is set to a negative value. Also, $\vec{x}_*$ will denote the best solution obtained so far. Specify $\epsilon > 0$.

2. For the ith decision variable, with $i = 1, 2 \ldots, I$, select a value $x(i)$ with probability equal to $p^k(i, x(i))$. Let the decision variable vector thus generated be denoted by $\vec{x}$. Evaluate (via simulation) the objective function associated with $\vec{x}$. Let the objective function value be G. Normalize this value, G, using:

$$F = \frac{G - G_{min}}{G_{max} - G_{min}}.$$

If $F > F_*$, set $\vec{x}_* = \vec{x}$ and $F^* = F$.

4. For $i = 1, 2, \ldots, I$, perform the following updates separately for each value of i:

For $a = 1, 2, \ldots, |\mathcal{A}(i)|$

4a. If $B(i,a) < B(i, x(i))$, set:

$$p^{k+1}(i,a) = p^k(i,a) - \mu \left[B(i,x(i)) - B(i,a)\right] p^k(i,a).$$

4b : If $B(i,a) > B(i, x(i))$, set

$$p^{k+1}(i,a) = p^k(i,a) + \mu \left[B(i,a) - B(i,x(i))\right] \left[1 - p^k(i,a)\right] \frac{p^k(i,x(i))}{|\mathcal{A}(i)| - 1}.$$

5. For every $i = 1, 2, \ldots, I$, perform the following update:

$$p^{k+1}(i,x(i)) = 1 - \sum_{a \neq x(i)} p^{k+1}(i,a).$$

6. For every $i = 1, 2, \ldots, I$, check if for *at least one* $a \in \mathcal{A}(i)$,

$$p^k(i, a) > 1 - \epsilon.$$

If the above is true for all values of i in $\{1, 2, \ldots, I\}$, then return $\vec{x^*}$ as the best solution and stop. Otherwise, go to Step 7.

7. For every $i = 1, 2, \ldots, I$, perform the following update:

- If $F > B(i, x(i))$, set $B(i, x(i)) = F$.

8. Set $k \leftarrow k + 1$ and return to Step 2.

In the following two sections, we will provide computational results obtained from empirical tests on large-scale stochastic discrete-optimization problems using LA.

3. Buffer Optimization in Transfer Lines

In this section, we discuss the empirical tests on a buffer-management problem commonly experienced in flow shops or transfer lines. In such systems, parts flow from one machine (or production stage) to another. When parts complete their processing on one machine (or stage), they are temporarily stored in between that machine and the next inside containers or "*buffers*." The Just-In-Time philosophy, which is an integral part of lean manufacturing, calls for placing an upper limit on the size (capacity) of these buffers (see [3] for a recent text on lean production systems). It is well-known [2] that for unreliable machines with random production times and random repair times, it becomes quite difficult to generate exact theoretical models for performance analysis — especially when the number of machines is large. A performance analysis model is a critical first step in optimizing buffer capacities. While an extensive literature exists on this topic (see [17] and [1]), much of it makes simplifying assumptions to develop tractable theoretical models. The problem was first studied by [9]. A subset of the more recent works include [44, 13, 41]. Simulation can be used effectively for generating a performance-analysis model. Thereafter, an AI-based tool like LA can be combined with the simulator for optimizing buffer capacities. We describe the problem in some detail next and then present some numerical results.

3.1 A problem description

Consider a linear sequence of machines (see Figure 1) in which one machine feeds parts to the next. In such machines, disruptions, arising out of differences in production rates or failures of machines, can lead to a significant reduction in the overall production rate of the line, i.e., the throughput. Hence, typically, buffers are placed in between machines to absorb the fluctuations in the line — fluctuations arise out of inequality of production rates between adjacent machines, and idle times of machines due to failure. Placing an upper limit on the buffer size is a notion that has origins in the concepts of Kanban and CONWIP. The idea works as follows: When a downstream buffer is full, the machine that feeds material into the buffer must stop. This is called *blocking*. When there are no parts in the buffer

preceding a machine, the machine has to stop until the preceding buffer acquires a part. This is called *starving*. Buffers can smoothen the activities on a production line thereby increasing — to some extent — the production rate, and hence the profit rate, of the line as whole. However, they represent locked-up capital and risk. Holding large buffers also results in increased inventory-holding costs. As a result, a commonly-experienced problem in buffer management is to find the values for the maximum buffer sizes that optimize overall profits. Our goal here is to compare the performance of LA to simulated annealing (SA), which is a standard meta-heuristic, on a large-scale version of the problem on which it is difficult to develop an exact theoretical model.

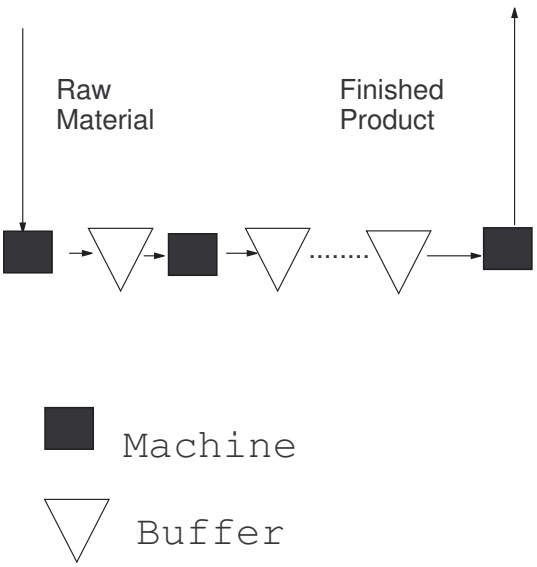

Figure 1: A transfer line with intermediate buffers.

We next provide some notation needed to develop the optimization model:

- n: the number of buffers in a system of $(n+1)$ machines,
- π: the profit per part produced,
- h: the WIP (work-in-progress) holding cost per unit piece per unit time,
- $\vec{b} = \{b_1, b_2, \ldots, b_n\}$: a vector in which b_i denotes the maximum size of the buffer between the ith and $(i+1)$th machine,
- $T(\vec{b})$: the expected throughput (parts produced per unit time by last machine) associated with the transfer-line system when the buffer vector is $\vec{b}$,
- $\bar{B}_i$: the average inventory in the ith buffer,
- α_i: the production rate of the ith machine for $i = 1, 2, \ldots, n+1$,
- β_i: the failure rate of the ith machine for $i = 1, 2, \ldots, n+1$, and

- γ_i: the repair rate of the ith machine for $i = 1, 2, \ldots, n+1$.

Note that the time between productions, failures, and repairs are random variables. Then a generic optimization problem is to maximize:

$$\rho = \pi T(\vec{b}) - h \sum_{i=1}^{n} \bar{B}_i,$$

such that $b_i \in \mathcal{N}$, the set of non-negative integers, and $i = 1, 2, \ldots, n$. It is, as mentioned above, quite difficult to develop closed-form expressions for the objective function under general assumptions such as arbitrary distributions for production times, failure time, and repair time. The tests presented below were conducted on large problems. We now describe the simulation-based model developed.

3.2 A simulation model

Consider a probability space $(\Omega, \mathcal{F}, P)$, where P denotes the distribution of the long run-cost of running the transfer-line system. The system can be simulated, and from the distribution P, one can generate a finite but large sequence of random samples $\{\omega^1, \omega^2, \ldots \omega^k\}$. Let $\theta(i, \omega^j, t)$ denote the number of units in the ith buffer at time t in the sample ω^j, and let $\Phi(\omega^j, \tau)$ denote the number of units produced by the last machine *by* time τ in the sample ω^j. Then, by the strong law of large numbers, with probability 1, the long-run expected profits can be estimated with a simulation model using the following:

$$\rho = \pi \lim_{k\to\infty} \frac{1}{k} \sum_{j=1}^{k} \lim_{\tau\to\infty} \frac{\Phi(\omega^j, \tau)}{\tau} - h \sum_{i=1}^{n} \lim_{k\to\infty} \frac{1}{k} \sum_{j=1}^{k} \lim_{\tau\to\infty} \frac{\int_0^\tau \theta(i, \omega^j, t)dt}{\tau}.$$

3.3 Empirical Results

We ran several tests on the buffer-management problem. System parameters for our tests and the average profits of LA and SA are presented in Table 2. In each performance evaluation, the system was run for a time of $100,000$ time units, and 4 replications were used for estimating the expected profit. The solution space of the largest problem is approximately 50^{10}, and as such exhaustive search is ruled out. The production, repair, and failure arrivals are Poisson processes.

A number of different strategies were tried for SA. We omit a description of SA because it can be found in a wide variety of texts, e.g., see pages 13-14 of Pham and Karaboga [37]. The first issue in selecting a strategy for using SA is that of generating neighbors. The strategy we followed worked as follows. Given a solution $(b_1, b_2, \ldots, b_n)$, the next solution is: $(b_1 + z_1, b_2 + z_2, \ldots, b_n + z_n)$ where z_i for $i = 1, 2, \ldots, n$ is a random number from the uniform discrete distribution assuming the value of 0 with probability equaling 0.5 and the value of 1 with the same probability. The second issue in SA is that of the exploration probability which is guided by the temperature. The temperature, T, in the mth iteration was set at $10000/ln(1+m)$ in our experiments using the rule in [24]. A newly generated worse solution was accepted with a probability of $exp(-\Delta/T)$ where Δ denotes the positive difference between the objective function value (cost) of the new (worse) solution

Table 2: A comparison of average profits obtained with LA and SA on the buffer-optimization problem. For all the cases, $\pi = 20$.

n	$(\alpha_i, \gamma_i, \beta_i), i = 1, 2, \ldots, n$	h	LA	SA
4	(3.7,0.17,0.07)(1.5,0.37,0.11)(1.1,0.78,0.49)(3.0,0.5,0.19)	1	8.35	8.31
4	(3.0,0.45,0.38)(1.0,0.55,0.3)(2.0,0.5,0.35)(3.6,0.4,0.45)	1	7.88	7.88
5	(1.2,0.3,0.1)(1.0,0.5,0.3)(3.0,0.02,0.50)(3.0,0.3,0.4)(1.8,0.1,0.2)	1	1.87	1.88
5	(2.8,0.64,0.3)(1.7,0.83,0.4)(2.5,0.75,0.45)(3.4,0.85,0.35)(1.9,0.74,0.1)	1	10.00	9.99
5	(2.6,0.4,0.08)(3.0,0.4,0.24)(3.4,0.6,0.2)(4.7,0.5,0.17)(1.5,0.3,0.1)	1	17.76	17.70
6	(3,0.2,0.3)(1.0,0.5,0.5)(1.2,0.3,0.1)(1.8,0.1,0.2)(1.5,0.2,0.3) (2.0,0.3,0.4)	0.2	4.82	4.84
8	(1.0,0.52,0.25)(3.6,0.48,0.18)(1.7,0.58,0.23)(1.4,0.5,0.32) (2.8,0.47,0.19)(2.7,0.46,0.35)(1.6,0.66,0.26)(1.2,0.41,0.20)	0.8	3.88	3.87
9	(2.5,0.7,0.2)(1.5,0.6,0.1)(2.8,0.8,0.3)(3.6,0.8,0.2)(2.1,0.7,0.1) (1.9,0.6,0.1)(2.7,0.8,0.3)(3.0,0.5,0.2)(2.0,0.6,0.3)	0.3	15.56	15.56
10	(2.7,0.8,0.1)(1.8,0.3,0.15)(2.1,0.5,0.3)(2.3,0.5,0.2)(1.6,0.8,0.27) (2.7,0.7,0.3)(1.5,0.6,0.12)(1.5,0.6,0.2)(1.2,0.6,0.13)(2.6,0.4,0.3)	0.1	13.81	13.79
10	(2.4,0.465,0.365)(1.7,0.565,0.215)(2.8,0.485,0.305)(2.2,0.455,0.375) (2.1,0.455,0.340)(2.5,0.390,0.390)(1.1,0.5,0.265)(1.3,0.49,0.285) (1.6,0.495,0.255)(0.8,0.505,0.240)	0.1	6.7041	6.24

and the same for the current solution. The results obtained are also provided in Table 2. A positive conclusion from our results is that LA's performance is comparable to a standard meta-heuristic, i.e., SA. LA has been shown to converge to near-optimal solutions on small problems [52].

4. Airline Revenue Management

A central problem in airline revenue maximization is that of *seat allocation*, i.e., selecting the optimal mix of passengers from the pool available. Generally, on most flights, seats are offered at many *different* fares because there are differences in passenger expectations. Many business passengers are willing to pay higher fares for additional features, while others are not. As a result, it makes business sense to reserve a few seats for high-revenue passengers who usually make reservations towards the end of the booking horizon. Passengers who pay the same fare are said to belong to the same *fare class.* The decision-making problem is centered on the following question: *What should the upper limits on the number of seats to be sold in each fare class be?* Setting the wrong limits can result in a situation in which there are no seats remaining for higher-fare passengers, who generally come later in the booking horizon. On the other hand, if all seats are reserved for higher fares, then most passengers will go elsewhere resulting in lowered revenues for the carrier. We now provide a more detailed description of the problem.

4.1 A problem description

Here, we consider a single leg of a journey. In each leg, the goal is to determine the number of passengers who can be allowed to buy tickets at a given fare. The problem is complicated by uncertainties in customer behavior: customer arrival is a random process and some passengers cancel tickets. As a result, airlines *overbook* planes in order to take care of last-minute cancellations; otherwise, planes can fly with empty seats, which adds to the complexity of the problem because cancellations are random. Fare classes are typically determined by the time of arrival in the booking horizon, the total number of flights for the passenger, etc. We do not concern ourselves with how fare classes are determined because our simulation-based model will work for any type of classification.

Real-world airline systems are characterized by the following features: (i) random customer arrivals for booking, (ii) random cancellations, (iii) non-sequential arrival patterns, i.e., there is no particular order of arrival of passengers according to their fare classes (in sequential arrivals, low fare classes arrive first), and (iv) no-shows at departure time. Most theoretical models in the literature *cannot* accommodate all these assumptions simultaneously because that leads to intractable models — especially when one considers concurrent arrival patterns and general cancellation probabilities [46, 56]. This makes the use of *simulation-based* models attractive. A large number of researchers both in the academic and practitioner world have worked on this problem. A literature review of the problem reveals the following works: [31, 43, 5, 12, 58, 30, 8, 38, 46, 55, 50, 6, 56, 22, 20]. Some of these works use a "network version" of the problem; the network version can usually be decomposed into a number of independent single-leg problems.

In our simulation model, we assume concurrent or non-sequential arrival patterns, that cancellations can occur, and that cancellation rates are independent for every fare class. We combine a single-leg airline simulator with an LA-based and an SA-based optimizer to determine the optimal limits on the fare classes.

4.2 A simulation model

Mathematically, the seat-allocation can be cast in the following form: find the value of the booking-limit vector $\vec{x} = \{x_1, x_2, \ldots, x_n\}$, where n is the number of fare classes, that maximizes the expected revenue from each flight, i.e.,

$$\max_{\vec{x}\in\theta} \mathsf{E}_P[T(\vec{x}, \omega^k)], \text{ in which}$$

$$\mathsf{E}_P[T(\vec{x}, \omega^k)] = \lim_{k\to\infty} \frac{T(\vec{x}, \omega^k)}{k}, \text{ with probability 1}$$

where P, a probability measure on a sample space $(\Omega, \mathcal{F})$, denotes the distribution of the revenue from a flight, E_P denotes expectation with respect to P, θ is a subset of $\mathcal{R}^n$, and $T : \mathcal{R}^n \times \Omega \to \mathcal{R}$ is a scalar-valued function such that $T(\vec{x}, \omega^k)$ denotes the total revenue obtained from the ω^kth sample flight, in which the booking limits vector is $\vec{x}$. In what follows, we drop the superscript k in ω^k to improve readability. Some more notation, needed to define $T(\vec{x}, \omega)$, is as follows:

- A_v: penalty incurred by passenger for cancellation in the vth class,
- f_v: revenue (fare) associated with the vth class where $f_1 < f_2 < \cdots < f_n$,
- B: the penalty incurred by company for bumping a passenger,
- C: plane's capacity, and
- H: the time horizon over which booking is permitted.

Also, if the booking-limits vector is $\vec{x}$, then $\rho_v(\vec{x}, \omega)$ will denote the number of passengers admitted in the vth class by the end of the ωth flight, and $c_v(\vec{x}, \omega)$ will denote the number of passengers who canceled tickets from the vth class by the end of the ωth flight. Then, the total revenue obtained in the ωth sample flight, if $\vec{x}$ is the booking-limits vector, can be expressed as:

$$\begin{aligned} T(\vec{x}, \omega) &= \sum_{v=1}^{n} \left(\rho_v(\vec{x}, \omega) - c_v(\vec{x}, \omega)\right) f_v + \sum_{v=1}^{n} c_v(\vec{x}, \omega) A_v \\ &\quad - B \max\left[0, \left\{\sum_{v=1}^{n} (\rho_v(\vec{x}, \omega) - c_v(\vec{x}, \omega)) - C\right\}\right]. \end{aligned}$$

In the right hand side of the above, the first term denotes the revenues obtained from the *net* sale of seats, the second denotes the revenues from cancellations, and the last denotes the net *cost* of disallowing boarding. The average revenue per unit time is then

$$\rho = \frac{\mathsf{E}_P[T(\vec{x}, \omega)]}{H}.$$

4.3 Empirical Results

In Table 3, we present the parameters for the single-leg-problem scenarios we studied. The average profits associated with LA, SA, and the EMSR (expected marginal seat revenue) heuristic [5] — a widely-used industrial heuristic — are presented in Table 4. We describe the EMSR heuristic in Appendix A. As is clear from Table 4, LA runs neck-to-neck with SA and outperforms EMSR. For SA, we used the EMSR solution as the starting solution. Also, the temperature in the mth iteration was set at $1000/ln(1+m)$. See [37] (pages 13-14) for a standard description of SA.

Table 3: Input parameters for the airline revenue management experiments. λ denotes the mean rate of Poisson arrivals, cd_i denotes the probability that the arriving passenger belongs to class i, and PC denotes the cancellation probability. $H = 100$

System	C	λ	(f_1, f_2, f_3, B)	(cd_1, cd_2, cd_3)	PC
1	100	1.00	(100,175,250,300)	(0.5,0.3,0.2)	0.10
2	100	1.00	(199,275,350,400)	(0.5,0.3,0.2)	0.15
3	100	1.00	(350,450,550,600)	(0.5,0.3,0.2)	0.20
4	100	1.50	(100,175,250,300)	(0.7,0.2,0.1)	0.10
5	100	1.5	(199,275,350,400)	(0.7,0.2,0.1)	0.15
6	100	1.5	(350,450,550,600)	(0.7,0.2,0.1)	0.20
7	150	2.0	(100,175,250,300)	(0.3,0.3,0.4)	0.10
8	200	2.0	(100,175,250,300)	(0.7,0.2,0.1)	0.10
9	200	2.0	(199,275,350,400)	(0.7,0.2,0.1)	0.15
10	200	2.0	(350,450,550,600)	(0.7,0.2,0.1)	0.20
11	150	2.0	(199,275,350,400)	(0.3,0.3,0.4)	0.0
12	200	2.0	(100,175,250,300)	(0.5,0.3,0.2)	0.0
13	100	2.0	(100,175,250,300)	(0.5,0.3,0.2)	0.10
14	100	2.0	(199,275,350,400)	(0.5,0.3,0.2)	0.15
15	100	2.0	(350,450,550,600)	(0.5,0.3,0.2)	0.20

5. Computational and convergence properties

A number of computational and analytical properties of the LA algorithm deserve special mention. We first discuss some computational properties. Many IT-based algorithms

Table 4: A comparison of LA's performance (with respect to average revenues ρ) with that of SA and EMSR on the revenue-management problem

System	LA	SA	EMSR
1	174.982	171.28	144.42
2	259.141	258.13	238.23
3	442.240	442.240	394.47
4	182.267	186.34	142.13
5	315.590	321.98	251.88
6	539.873	531.24	444.44
7	261.907	261.907	260.74
8	367.851	362.67	233.64
9	545.759	545.759	424.46
10	849.800	849.800	731.47
11	412.833	414.34	357.88
12	348.781	341.234	289.14
13	204.559	204.559	128.03
14	331.575	331.575	252.10
15	568.654	568.654	470.96

require large data-bases. It turns out that LA requires a maximum of $I \cdot \max_i |\mathcal{A}(i)|$ probabilities in its data base, but obviously this is much smaller than the size of the solution space, which is $\prod_{i=1}^{I} |\mathcal{A}(i)|$. An obvious problem that can be encountered in an LA-simulator combination is that the same solution may be evaluated repeatedly. This can be easily avoided by using binary trees as was done in [40].

Although asymptotic convergence of the algorithm has already been established in [51], a number of convergence properties of the algorithm are interesting in the context of *practical applications*, and deserve further analysis. In particular, what is often of interest in meta-heuristics is to generate a meaningful stopping criterion. The stopping criterion, presented in Step 6, needs to be analyzed. In practice, as has been documented in the original paper [51], the probabilities of selecting the optimal (or near-optimal) solution start converging to 1. In this context, we present Theorem 1. For Theorem 1, we will first need the following lemma.

Lemma 1. *The quantity $p^n(i, a)$ — generated in the algorithm — stays in the interval $]0, 1[$ (open interval between 0 and 1) for all values of $i \in \mathcal{S}$ and all values of $a \in \mathcal{A}(i)$ for any value of n.*

The proof is placed in Appendix B. Indeed, the need for the lemma may not be immediately obvious; in fact, it does *not* state an obvious fact. The algorithm uses the values defined in $p(.,.)$ and updates them. However, if they are to serve as *probabilities*, it is necessary to show that these values remain between 0 and 1; the algorithm's updating mechanism in Step 5 only ensures that the values for any decision variable sum to 1.

Essentially Theorem 1 will prove that once an optimal solution is struck, the probability of selecting the optimal solution will increase monotonically, and in practice this assures us that the algorithm will gradually settle down on the optimal solution. Therefore, once the algorithm visits the optimal solution, it will visit it repeatedly with increasing probability. In other words, the optimal solution is selected with an increasing probability after its first visit, which ensures that asymptotically, the algorithm gets "trapped" in the optimal solution.

Theorem 1: *The probability of selecting the optimal solution in the kth iteration, that is, $\prod_{i=1}^{I} p^k(i, x^*(i))$, increases monotonically with k — after the optimal solution is struck for the first time.*

The proof is placed in Appendix C. In practice, the usefulness of the above result can be exploited in setting a termination criterion. Since the optimal probabilities will increase after striking the optimal solution, the algorithm can be terminated when the probabilities exceed the threshold $1 - \epsilon$ set in Step 6.

6. Conclusions

This report presented an empirical study of a useful AI-based algorithm on two complex problems in management science — one from stochastic production lines and the other from airline revenue management. Both problems, under very general assumptions, defy exact theoretical analysis. As such, they form nice candidates for testing the efficacy of an IT-based algorithm such as LA. LA demonstrated that its performance is comparable to that of SA, which is a well-known meta-heuristic. We also identified and analyzed a stopping

criterion for the LA algorithm. Although extensive tests on other domains will be required before drawing definitive conclusions about the effectiveness, our empirical results indicate that LA appears to be a suitable candidate for solving large-scale discrete-optimization problems using AI- and IT-based paradigms.

The LA algorithm enjoys some properties that distinguish it from other meta-heuristics. It does not require a starting solution; also, it has a natural stopping criterion. On the downside, it admittedly has a significant memory burden. Like other meta-heuristics, it is computationally intensive, but this is less of an issue with the increasing power of computers, which has given birth to the field of information technology.

Our empirical tests were geared towards stochastic domains of interest in the industry, but the algorithm could be also used for solving traveling-salesman-type problems and facilities-designing problems. This may form an interesting avenue for further research. Also, the rate of its convergence and computational complexity were not analyzed, and these are important topics for future research.

Acknowledgements This research was supported in part by the National Science Foundation from Grant DMI- 0114007 to the first author.

References

[1] Altiok, T. (1996). *Performance Analysis of Manufacturing Systems.* Springer.

[2] Altiok, T. and S. Stidham, Jr. (1983). The allocation of inter-stage buffer capacities in production lines. *IIE Transactions.* **15**. pp. 292-299.

[3] Askin, R, and J. Goldberg. (2002). *Design and Analysis of Lean Production Systems.* John Wiley and Sons, NY, USA.

[4] Baba, N. (1984). *New Topics in Learning Automata Theory and Applications.* Lecture Notes in Control and Information Series. **71**. Springer Verlag, Berlin.

[5] Belobaba, P.P. (1989). Application of a probabilistic model to airline seat inventory control. *Operations Research.* **37**. pp. 183-197.

[6] Bertsimas, D. and S. de Boer. (2003). A stochastic booking-limit policy for airline network revenue management. *Working paper* at OR Center, MIT, Cambridge, MA.

[7] Bonnans, J.F. and Shapiro, A. (2000). *Perturbation Analysis of Optimization Problems*. Springer.

[8] Brumelle, S.L. and J.I. McGill. (1993). Airline seat allocation with multiple nested fare classes. *Operations Research.* **41.** pp. 127-137.

[9] Buzacott, J.A. (1967). Automatic Transfer lines with buffer stocks. *International Journal of Production Research.* **6**. pp. 183-200.

[10] Colorni, A., M. Dorigo, and V. Maniezzo (1992). Distributed optimization by ant colonies. In *Proceedings of the First European Conference on Artificial Life.* MIT Press, MA. pp 134-142.

[11] Congram, R.K., C.N. Potts, and S.L. van de Velde. (2002). An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing.* **14 (1)**. pp. 52-67.

[12] Curry, R.E. (1990). Optimal airline seat allocation with classes nested by origins and destinations. *Transportation Science.* **24.** pp. 193-204.

[13] Dallery, Y., R. David, and X.L. Xie. (1989). Approximate Analysis of transfer lines with unreliable machines and finite buffers. *IEEE Transactions on Automatic Control.* **34**(9). pp. 943-953.

[14] El-Fattah, Y.M. and C. Foulard. (1978). *Learning Systems:Decision, Simulation and Control.* Number 9 in Lecture Notes in Control and Information Sciences, Berlin: Springer-Verlag.

[15] Feo, T.A. and M.G. Resende. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization.* **6.** pp. 109-133.

[16] Frasconi, P., M. Gori, M. Maggini, and S.G. Soda. (1997). Representation of Finite State Automata in Recurrent Radial Basis Function Networks. *Machine Learning.* **23**. pp. 5-32.

[17] Gershwin, S.B. (1987). *Manufacturing Systems Engineering.* Prentice Hall, Englewood Cliffs, NJ.

[18] Glover, F. (1990). Tabu Search: A Tutorial.*Interfaces.* **20**. pp 74-84.

[19] Gosavi, A. (2003)*Simulation-based Optimization: Parametric Optimization Techniques and Reinforcement Learning.* Kluwer Academic, MA.

[20] Gosavi, A., N. Bandla, and T.K. Das (2002). A Reinforcement Learning Approach to Airline Seat Allocation for Multiple Fare Classes with Overbooking" IIE Transactions, **34**, pp 729-742.

[21] Gosavi, A., T.K. Das, and S. Sarkar. (2004). A simulation-based learning automata approach to solve semi-Markov average reward problems. *IIE Transactions,* 36, pp 1-11

[22] Gosavi, A. (2004). A Reinforcement Learning Algorithm Based on Policy Iteration For Average Reward: Empirical Results with Yield Management and Convergence Analysis, *Machine Learning,* 55(1), pp 5-29.

[23] Gosavi, A. (2004). Reinforcement Learning for long-run average cost, *European Journal of Operational Research,* 155, pp 654-674.

[24] Hajek, B. (1988). Cooling schedules for optimal annealing. *Mathematics of Operations Research.* **13**, pp 311-329.

[25] Haykin, S. *Neural Networks: A Comprehensive Foundation.*McMillan, New York, NY, USA, 1994.

[26] Ho, Y.C., R. Sreenivasan, and P. Vakili. (1992). *Ordinal optimization of Discrete-Event Dynamic Systems: Theory and Applications.* Kluwer Academic, MA.

[27] Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press. Ann Arbor.

[28] Kirkpatrick, S., C.D. Gellat and M.P. Vecchi. (1983) Optimization by Simulated Annealing. *Science.* **220**. pp. 671-680.

[29] Lakshmivarahan, S. (1981) *Learning Algorithms: Theory and applications.* Springer-Verlag, NY.

[30] Lee, T.C. and M. Hersh. (1993) A model for dynamic airline seat inventory control with multiple seat bookings. *Transportation Science.* **27.** pp. 252-265.

[31] Littlewood, K. (1972). Forecasting and control of passenger bookings. In *Proceedings of the AGIFORS Symposium.* pp. 95-117.

[32] Metropolis, N., A. Rosenbluth, N. Rosenbluth, A. Teller, and E. Teller. (1953). Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**. pp. 1087-1092.

[33] Narendra, K. and M.A.L. Thathachar. (1989). *Learning Automata: An Introduction.* Prentice-Hall. Englewood Cliffs, NJ.

[34] Norman, M. (1972) *Markov Processes and Learning Models*, Academic Press, NY, USA.

[35] Oommen, B.J., and T.D. Roberts. (2000). Continuous Learning Automata Solutions to the Capacity Assignment Problem. *IEEE Transactions on Computers.* **49(6)**. pp. 608-620.

[36] Ozden, M. and Y-C. Ho. (2003). A probabilistic solution-generator for simulation. *European Journal of Operational Research,* 146, pp 35-51.

[37] Pham, D. and D. Karaboga. (2000) Intelligent Optimisation Techniques. Springer, London, UK.

[38] Robinson, L.W. (1995). Optimal and approximate control policies for airline booking with sequential non-monotonic fare classes. *Operations Research.* **43.** pp. 252-263.

[39] Rummery, G. and M. Niranjan. (1994) On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University.

[40] Sarkar, S. and S. Chavali. (2000). Modeling Parameter Space Behavior of Vision Systems Using Bayesian Networks. *Computer Vision and Image Understanding.* **79.** pp. 185-223.

[41] Shi, L. and S. Men (2003). Optimal buffer allocation in production lines, *IIE Transactions*, **35**. pp. 1-10.

[42] Shi, L. and S. Olafsson (2000). Nested Partitions Method for Global Optimization, *Operations Research*, **48**. Pp. 390-407.

[43] Shlifer, E. and Y. Vardi. (1975). An airline overbooking policy. *Transportation Science.* **9**. Pp. 101-114.

[44] So, K.C. (1997). Optimal buffer allocation strategy for minimizing work-in-process inventory in unpaced production lines. *IIE Transactions.* **29**. pp. 81-88.

[45] Spall, J.C. (1992). Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. *IEEE Transactions on Automatic Control.* **37.** pp 332-341.

[46] Subramaniam, J., S. Stidham, Jr., and C.J. Lautenbacher. (1999). Airline yield management with overbooking, cancellations, and no-shows. *Transportation Science.* **33**. pp. 147-167.

[47] Sutton, R. (1996) Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding, In *Advances in Neural Information Processing Systems 8* MIT Press, pages 1038-1044, Cambridge, MA.

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

[49] Taillard, E.D., L.M. Gambardella, M. Gendreau, and J-Y. Potvin. (2001). Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operational Research.* **135.** pp. 1-16.

[50] Talluri, K.T. and G. van Ryzin. (1999). An analysis of bid-price controls for network revenue management. *Management Science.* **44.** pp. 1577-1593.

[51] Thathachar, M.A.L. and P.S. Sastry. (1987). Learning Optimal Discriminant Functions Through a Cooperative Game of Automata. *IEEE Transactions on Systems, Man, and Cybernetics.* **17**. pp. 73-85.

[52] Tezcan, T. and A. Gosavi. (2001). Optimal buffer allocation in production lines using an automata search. *Proceedings of the 2001 IERC,* Dallas, TX.

[53] Tsetlin, M.L. (1973). *Automata Theory and Modeling of Biological Systems.* Academic Publishers, NY.

[54] Unsal, C., P. Kachroo, and J.S. Bay. (1999). Multiple Stochastic Learning Automata for Vehicle Path Control in an Automated Highway System. *IEEE Transactions on Systems, Man, and Cybernetics.* **29.** pp. 120-129.

[55] van Ryzin, G. and J.I. McGill. (2000). Revenue management without forecasting or optimization: An adaptive algorithm for determining seat protection levels. *Management Science.* **46.** pp. 568-573.

[56] van Ryzin, G. and G. Vulcano. (2003). Simulation-based optimization of virtual nesting controls of network revenue management. Working paper series at Columbia University, NY.

[57] Watkins, C. (1989). *Learning from delayed rewards*, Ph.D. thesis, King's college, Cambridge, England.

[58] Wollmer, R.D. (1992). An airline seat management model for a single route when lower fare classes book first. *Operations Research.* **40.** pp. 26-37.

[59] Yan, D., and H. Mukai. (1992). Stochastic Discrete Optimization. *SIAM Journal of Control and Optimization.* **30.** pp. 594-612.

APPENDIX

Appendix A. EMSR

The Expected Marginal Seat Revenue (EMSR) rule of Belobaba [5] has its origins in the rule of Littlewood [31]. It forms a very simple heuristic rule that works very efficiently for seat-allocation. Let n denote the number of fare classes. Also, our numbering scheme will imply that if $i > j$, $f_i > f_j$. Let $S(i,j)$ denote the number of seats to be *protected* from class j for higher class i. Then according to Littlewood's rule:

$$Pr(X_i > S(i,j)) = \frac{f_j}{f_i}, \text{ for } j = 1, 2, \ldots, n-1, \quad i = j+1, j+2, \ldots, n, \tag{1}$$

where X_i denotes the expected number of requests that will arrive for class i in the total booking horizon. Using the demand-arrival rates and probabilities, one can compute the values of $S(i,j)$ via Equation (1). Then, the value of $x(j)$, the booking limit for the jth fare class, is given as:

$$x(j) = C - \sum_{i>j} S(i,j).$$

Appendix B. Proof of Lemma 1

Proof The proof will be by induction. We will first prove that the result is true for $n = 1$. Since $p^1(i,a) = 1/|\mathcal{A}(i)|$, for all (i,a), and since no updating occurs when $n = 1$, the quantities stay in the interval $]0,1[$ (i.e., open interval between 0 and 1).

Let us assume that the result is true when $n = m$. Hence $p^m(i,a) \in (0,1)$ for all (i,a) pairs. There are three possible types of updates: Case 1 (update in Step 4a), Case 2 (update in Step 4b), and Case 3 (update in Step 5). We consider these on a case-by-case basis.

Case 1: (Step 4a update) Note that $\mu \in]0,1[$. Here:

$$p^{m+1}(i,a) = p^m(i,a)[1 - \mu\{B(i,x(i)) - B(i,a)\}].$$

Since $B(i,x(i)) > B(i,a)$, we have that $\mu\{B(i,x(i)) - B(i,a)\}$ lies in the interval $]0,1[$, which implies that $[1 - \mu\{B(i,x(i)) - B(i,a)\}]$ lies in the interval $]0,1[$. Then, since both $p^m(i,a)$ and $[1 - \mu\{B(i,x(i)) - B(i,a)\}]$ lie in the interval $]0,1[$, $p^{m+1}(i,a)$, too, must lie in the interval $]0,1[$ for this update.

Case 2: (Step 4b update) Note that each of the following four quantities lie in the interval $]0,1[$: μ, $[B(i,a) - B(i,x(i)]$, $p^m(i,x(i))$, and $1/[|\mathcal{A}(i)| - 1]$. Hence, their product must also lie in the open interval $]0,1[$. Let us denote the product by L. Then,

$$\begin{aligned} p^{m+1}(i,a) &= p^m(i,a) + L(1 - p^m(i,a)) \\ &< p^m(i,a) + (1 - p^m(i,a)) \\ &= 1 \end{aligned}$$

The above implies that $p^{m+1}(i,a) < 1$. Also,

$$\begin{aligned} p^{m+1}(i,a) &= p^m(i,a) + L(1 - p^m(i,a)) \\ &> 0 \text{ (note that } p^m(i,a) > 0 \text{ and } L > 0). \end{aligned}$$

From the above, $p^{m+1}(i,a)$, for this update, must lie in the interval $]0,1[$.

Case 3:(Step 5 update) From the analysis in Step 4a and Step 4b, we have that for $a \neq x(i)$, $p^{m+1}(i,a)$ is strictly greater than 0. Then from Step 5, it is clear that: $p^{m+1}(i,x(i))$ is less than 1. To show that $p^{m+1}(i,x(i))$ is strictly greater than 0, we need to express Step 5 in the following form.

$$p^{m+1}(i,x(i)) = p^m(i,x(i)) + D, \tag{2}$$

where the sign of D depends on the updates that occur in Steps 4a and 4b. We will now set up a general expression for D. Let us define $\mathcal{S}_1$ to be the set of values that were updated in the mth iteration using Step 4a, and $\mathcal{S}_2$ to be the set of values that were updated in the mth iteration using Step 4b. Then:

$$D = \sum_{a \in \mathcal{S}_1} \mu[B(i,x(i)) - B(i,a)]p^m(i,a) - \sum_{a \in \mathcal{S}_2} \mu[B(i,a) - B(i,x(i))]p^m(i,x(i))\frac{1-p^m(i,a)}{|\mathcal{A}(i)|-1}. \tag{3}$$

Note that with this expression for D, Equation (2) is equivalent to Step 5. Now, if $D \geq 0$, then from Equation (2), we have that:

$$p^{m+1}(i,x(i)) \geq p^m(i,x(i)) > 0,$$

and hence $p^{m+1}(i,a)$ is strictly greater than 0. However, if $D < 0$, some more work is needed. Let us define L' as follows:

$$L' = \mu \max_{a \in \mathcal{S}_2}[B(i,a) - B(i,x(i))] \max_{a \in \mathcal{S}_2}[1 - p^m(i,a)].$$

Then, by its definition, $0 < L' < 1$.

Now,

$$\begin{aligned}
D &= \sum_{a \in \mathcal{S}_1} \mu[B(i,x(i)) - B(i,a)]p^m(i,a) \\
&\quad - \sum_{a \in \mathcal{S}_2} \mu[B(i,a) - B(i,x(i))]p^m(i,x(i))\frac{1-p^m(i,a)}{|\mathcal{A}(i)|-1} \\
&\geq - \sum_{a \in \mathcal{S}_2} \mu[B(i,a) - B(i,x(i))]p^m(i,x(i))\frac{1-p^m(i,a)}{|\mathcal{A}(i)|-1} \\
&= -\mu p^m(i,x(i)) \sum_{a \in \mathcal{S}_2} [B(i,a) - B(i,x(i))]\frac{1-p^m(i,a)}{|\mathcal{A}(i)|-1} \\
&\geq -\mu p^m(i,x(i)) \max_{a \in \mathcal{S}_2}[B(i,a) - B(i,x(i))] \max_{a \in \mathcal{S}_2}[1 - p^m(i,a)] \sum_{a \in \mathcal{S}_2} \frac{1}{|\mathcal{A}(i)|-1} \\
&= -p^m(i,x(i))L'\frac{1}{|\mathcal{A}(i)|-1} \sum_{a \in \mathcal{S}_2} 1 \\
&> -p^m(i,x(i))\frac{1}{|\mathcal{A}(i)|-1} \sum_{a \in \mathcal{S}_2} 1 \text{ (since } L' > 0) \\
&> -p^m(i,x(i)) \cdot \frac{1}{|\mathcal{A}(i)|-1} \max \left[\sum_{a \in \mathcal{S}_2} 1 \right]
\end{aligned}$$

$$= \quad -p^m(i, x(i)) \cdot 1 \text{ (since } \max|\mathcal{S}_2| = |\mathcal{A}(i)| - 1).$$

Thus, $D > -p^m(i, x(i))$, which implies that $p^m(i, x(i)) + D > 0$. Then, from Equation (2), $p^{m+1}(i, x(i)) > 0$. ∎ ■

Appendix C. Proof of Theorem 1

Proof It follows from Lemma 1 that any $p^k(i, a)$ is a probability. Note that in any iteration, for any given j, where $j = 1, 2, \ldots, I$, either $x(j) = x^*(j)$ or else $x(j) \neq x^*(j)$. After the optimal solution is struck, since we are maximizing the objective function value,

$$B(i, x^*(j)) \geq B(i, x(j))$$

for every $x(j) \neq x^*(j)$. Without loss of generality let us assume j to be i.
If $x(i) \neq x^*(i)$, then since $B(i, x^*(i)) \geq B(i, x(i))$, $p^k(i, x^*(i))$ will be updated in Step 4b, and hence $p^{k+1}(i, x^*(i)) \geq p^k(i, x^*(i))$.
If $x(i) = x^*(i)$, then the update for $p^k(i, x^*(i))$ is in Step 5. Furthermore, no value will be updated in Step 4b. Hence, we have that $\mathcal{S}_2$ in (3) will be an empty set. This implies that $D > 0$. Therefore from (2), we have that $p^{k+1}(i, x^*(i)) > p^k(i, x^*(i))$. ∎ ■