Engineering Journal issn: 0125-8281

Article

Optimal Assignment of Customer-Desired Items to the Fetching Robots in Superstores

Ali Nasir[1], **Muhammad Saadi**[1,*], **Rida Gelani**[1], **and Faisal Mustafa**[2]

1 Department of Electrical Engineering, Faculty of Engineering, University of Central Punjab, Lahore, Pakistan
2 Faculty of Management Sciences, University of Central Punjab, Lahore, Pakistan
E-mail: *muhammad.saadi@ucp.edu.pk (Corresponding author)

Abstract. This paper discusses a task assignment problem. The scenario under consideration is a superstore with a team of fetching robots. There is a set of customers each requiring a unique set of items. The goal is to assign the task of fetching the items to the available robots in such a way that the time and effort required for fetching the item is minimized. For this purpose, a Markov Decision Process based model has been proposed. The proposed-model is solvable using stochastic dynamic programming algorithms such as value iteration for the calculation of optimal task assignment policy. The analysis of the characteristics of the resulting optimal policy has been presented with the help of a numerical case study.

Keywords: Markov decision processes, task assignment problem, optimality criteria, shopping assistance.

ENGINEERING JOURNAL Volume 25 Issue 11
Received 12 February 2021
Accepted 21 November 2021
Published 30 November 2021
Online at https://engj.org/
DOI:10.4186/ej.2021.25.11.1

DOI:10.4186/ej.2021.25.11.1

1. Introduction

Shopping at a superstore is a routine activity for the urban population. While large superstores provide the facility of "everything at one place", the drawback is the increase in amount of effort required to find one particular thing. Considering the elderly and differently abled persons, it is important to find a way to facilitate shopping activity. Assuming that the shopping assistance is carried out using the fetching robots, there are four aspects of the shopping assistance problem, i.e., 1) Design and development of robotic manipulators for fetching the items, 2) Design of path planning algorithms for the fetching robots, 3) Design of the control algorithms for reference tracking in the motion of the fetching robots, 4) Design of the task assignment algorithms for deciding which robot shall fetch which of the desired items. In this regard, significant research has been carried out. Some hardware platforms have been developed [1, 2] and different solutions have been presented to the problem of automated shopping assistance [3, 4]. In this paper, we focus on the fourth aspect of the problem, i.e., task assignment for fetching the items.

In terms of shopping assistance via robotic manipulators, there have been many approaches. For example, in terms of hardware platforms, TOOMAS [1] and Shopbot [4] are prominent. Shopping assistance solution for blind people is presented in [5] and [2] whereas an approach to support elderly people is discussed in [6]. In terms of algorithms, a localization technique is discussed in [7]. A robotic hand mechanism for grasping fresh food in a supermarket is presented in [8]. A remotely operated shopping robot system is discussed in [9]. An automated human-guided shopping trolley is proposed in [10]. Other interesting relevant work includes [11-14].

Regarding the algorithms for multiple agents shopping assistance, there is not much work available in the literature. Two of the most recent references include [15] and [16]. None of these works talk about the shopping assistance as a team task or for that matter any interaction among multiple shopping assistance (intelligent) agents. On the other hand, human-robot or human-computer interaction is something that has attracted the attention of the researchers recently [17, 18].

The least addressed aspect of the multiagent shopping assistance via fetching robots is that of task assignment. Task assignment in general is a well-researched area. For example, a linear programming-based approach for task assignment is proposed in [19]. An approach for assignment of multi-period multi-site assignment problem is proposed in [20]. A two-step heuristics-based algorithm using Tabu search has been discussed in [21]. The design of warehouse is also important for autonomous fetching/placement. Such issues are discussed in [22] and [23]. More approaches on task assignment are presented in [24] and [25]. But among all of the proposed approaches, there is none that can be directly adopted for solving the problem of fetching task assignment discussed in this paper mainly because these methods do not incorporate uncertainty. Especially, given the stochastic nature of the problem as the arrival of a customer at the store and the number and type of items that are required by each customer are random phenomena, it is desirable to develop a task assignment model that incorporates the uncertainties. Fortunately, there are some sequential decision making tools that can be utilized for this purpose. One such tool is stochastic dynamic programming [26]. But in order to be able to use this tool, the problem of task assignment must be formulated as a Markov Decision Process (MDP).

An MDP models a sequential decision making problem as a tuple (S, A, R, T, γ) where S is a set of states, A is a set of actions, R is the state dependent reward function, T is state and action dependent transition function, and γ is discount factor (or depreciation factor for state rewards) ranging from 0 to 1 (excluding the boundary values 0 and 1). There are many ways to solve an MDP problem for an optimal policy (optimal with respect to expected discounted reward). For example, value iteration, policy iteration, linear programming, dynamic Bayesian networks and so on [26, 27].

Given the above discussion, major contribution of this paper is the formulation of an MDP model that can be solved using stochastic dynamic programing to obtain optimal task assignment policy for fetching robots. Specifically, we have defined the state variables for the task assignment problem that include the maximum item carrying capacity of each robot, the priority level of each customer, the existing assignment of items to be fetched by each robot, and the location of each robot. In order to enable the calculation of an optimal task assignment decision, a reward function has been proposed that is used to encourage the assignment of items in such a manner that no customer-desired items are left unassigned and the assignment is carried out to ensure minimum possible distance traveled by the robots while meeting the constraint of a maximum item carrying capacity. Furthermore, the transition probability function has been proposed for the state variables. Other contributions of the paper include discussion on the scalability of the proposed model and simulation based case study that presents various insights regarding the optimal policy calculated using the proposed model. Furthermore, the solution methodology involving value iteration for the proposed model has also been discussed in the paper.

2. Problem Formulation and Associated Assumptions

In this section we setup the stochastic sequential decision-making problem to be addressed. Assume that there is a big superstore with multiple aisles configured in parallel (let p be the number of aisles). We have a team of n robots ($p \geq n$) that are to assist m customers. Furthermore, each customer desires to buy i_j items where i is a list (of items) and subscript j represents the identity of the customer ($j \in \{1,2,\ldots,m\}$). There is a supervisor

(centralized computer) that can communicate with each robot and can assign items to the robots. Assignment of an item to the robot means that the robot is responsible to fetch that particular item. There are total of k items in the store (with respect to type) and location of each item (in terms of aisle number) is known. Assume that the map of the superstore and location of items is known to the robots. Also the robots are capable of determining their own location and send/receive the same to other robots (or to the supervisor). Finally, each robot is capable of grasping and carrying q items.

Given the above assumptions, the problem is to assign the items desired by the customers to the robots in such a way that is optimal with respect to the time and effort required to fetch the item to the customers.

The depiction of the problem setup is shown in Fig. 1 where four robots are indicated in four aisles of a superstore. The location of the supervisor is accessible to all of the robots. Moreover, the robots are assumed to be moveable from one aisle to another.

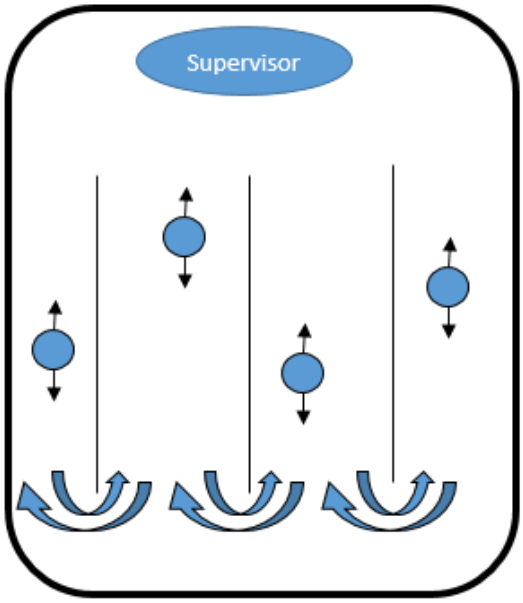

Fig. 1. Depiction of the Problem Setup.

It is important to note that in the scenario generated above, any customer arriving at the superstore provides the list of its desired items to the supervisor (which is a centralized computer system). The supervisor in turn has to assign the items to the operational robots so that the same could be fetched at the location of the supervisor (which is the same as the location of the customer).

3. Proposed MDP Model

In this section, we develop an MDP model that can be used to address the problem defined in the previous section. Specifically, we define the state variables involved in the problems, the reward function for optimization purposes, the state transition function in order to account for the uncertainties involved in the problem, the set of decisions, and the value of the discount factor for the problem.

3.1. Formulation of State Space

In order to formulate the state space, need to consider the variables in the problem that are needed for informed decision making. The variables have to be assigned carefully otherwise the problem would become overly complex. As mentioned in the previous section, our goal is to assign the items to the robots (for fetching purposes). The dynamics of the robots shall not be discussed here because we are not concerned with the control design or path planning. To assign the items for fetching, we must have knowledge about the items that are desired by the customers at any given time. Since we have assumed m customers and k items, there shall be $m \times k$ binary variables ($d^{i,j}$) in the state space indicating whether an item i is desired by customer j or not. Next, we need to have the information regarding the number of items assigned to a robot at any given time so that the item carrying capacity constraint for each robot is not violated. Since there are n robots, we need n variables representing the number of items assigned to a robot at any given time. Each of these variables can assume values between 0 and q (recall that q is the maximum number of items that a robot can carry). In order to ensure *first-come-first-served* policy, we may also have the priority level for each customer in our state space. Finally, to be able to minimize the distance to be traveled by the robot for fetching an item, the location of each robot must be part of the state space. We simplify the location in terms of aisles. Since there are p aisles, a robot can have location between 1 and p at any given time. The resulting state space is presented below:

$$
\begin{aligned}
&S = \{s_1, s_2, \dots s_z\},\\
&s_x = \{L_x, D_x, C_x, O_x\},\\
&L_x = \{l_x^1, l_x^2, \dots, l_x^m\},\\
&l_x^y \in \{1,2, \dots, m\},\\
&D_x = \{d_x^{1,1}, d_x^{1,2}, \dots, d_x^{k,m}\},\\
&d_x^y \in \{0,1\},\\
&C_x = \{c_x^1, c_x^2, \dots, c_x^n\},\\
&c_x^y \in \{1,2, \dots, q\}\\
&O_x = \{o_x^1, o_x^2, \dots, o_x^n\},\\
&\quad o_x^y \in \{1,2, \dots, p\}.
\end{aligned}
\tag{1}
$$

Here, L is the set of variables representing the priority level of each customer (e.g., which customer came earlier versus which came later). There are a total of m customers and the priority level of y^{th} customer in state $s_x(l_x^y)$ can range between 1 and m where 1 signifies highest priority and m signifies lowest. D is the set of variables representing whether a particular item is desired by a particular customer has been assigned to a robot ($d = 0$) or not ($d = 1$). There are total k items and m customers therefore $m \times k$ binary variables in D. C is the set of variables representing number of items assigned to a particular robot. There are n robots and each robot can

carry up to q items. Variable O represents the location of each robot. There are n robots and the location of each robot can vary between 1 and p (where p is the number of aisles).

3.2. Actions, Reward, and Transitions

Once the state space is defined, the next task is to identify the set of actions, reward function, and the state transition probability function. The set of actions for a supervisor in the superstore can be laid out as

$$A = \{a_{1,1,1}, a_{1,1,2}, \dots, a_{p,m,n}\}. \quad (2)$$

Here each action corresponds to assignment of a specific item (desired by a specific customer) to a specific robot. Since there are total p items, m customers, and n robots, the number of actions is $p \times m \times n$. Since we have modeled this problem as a sequential decision making, the supervisor can assign items to the robots one by one. For assignment of multiple items at a time, we could define a single action that assigns two items to two robots. But this would require adding actions for each possible pair of items assigned to each possible pair of robots and desired by each possible pair of customers. This would result in a large action space. Therefore, we avoid such formulation of actions in this paper.

In terms of reward function, there are many possible formulations. A linear formulation of the reward function is as follows:

$$R(s,a) = \alpha_1 \left(p.m^2 - \sum_{y_1=1}^{p} \sum_{y_2=1}^{m} l^{y_2}(s) d^{y_1,y_2}(s) \right) + \alpha_2 \left(p - \left| o^y(s) - x(a_{x,*,y}) \right| \right) \quad (3)$$

Here the reward function is a linear combination of two terms with positive weighting factors α_1 and α_2. The first term penalizes any unassigned items desired by the customer multiplied by the priority level of the customer. Since this is a reward function (as opposed to a cost function). The penalty is represented in terms of the reduction in the reward function. Note that the total number of desirable items by all customers is $p \times m$. Also, once an item is desired (not yet assigned) the corresponding d variable attains value of 1. As soon as the item is assigned to a robot, the corresponding d variable attains the value 0. The second term of the reward function represents the penalty on the assignment of an item to the far away robot. Note that the largest possible distance from the current location of the y^{th} robot in the state s ($o^y(s)$) to the location x of the item being assigned to the y^{th} robot is $(p-1)$. Note that $a_{x,*,y}$ refers to the action that assigns item x (located in aisle number x) to robot number y and the item could be desired by any customer (represented by "$*$"). With positive weighting functions, the reward function defined in (3) is guaranteed to be positive. This is desirable since a negative reward may result in unexpected behavior of the optimal control policy (or item assignment policy).

Other possible formulations of the reward function are exponential, logarithmic, or quadratic etc. For example, an exponential reward function is given as

$$\begin{aligned} R(s,a) &= \alpha_1 e^{f_1(s)} + \alpha_2 e^{f_2(s,a)} \\ f_1(s) &= \left(p.m^2 - \sum_{y_1=1}^{p} \sum_{y_2=1}^{m} l^{y_2}(s) d^{y_1,y_2}(s) \right) \\ f_2(s,a) &= \left(p - \left| o^y(s) - x(a_{x,*,y}) \right| \right) \end{aligned} \quad (4)$$

Notice that the reward function terms are essentially the same as in (4) but now instead of linear combination, (4) represents the combination of the exponentials. This may yield to different solution compared to that with (3). Which one is better is a discussion that is nontrivial. Opposite to the exponential formulation is the logarithmic formulation. We present it as

$$R(s,a) = \alpha_1 \log_{10}\left(1 + f_1(s)\right) + \alpha_2 \log_{10}(1 + f_2(s,a)) \quad (5)$$

Here, the term "1" is added to the arguments to keep the reward function positive (note that both f_1 and f_2 are positive semidefinite). A commonality in the reward functions (3)-(5) is that the two terms have been kept separate. We can have reward function where the two terms are multiplied. We present this case for only the linear reward and leave the rest to the reader for exploration.

$$R(s,a) = \alpha f_1(s) f_2(s,a) \quad (6)$$

It is clear that there might be infinitely many ways of formulating a reward function. Important issue here is to study and compare the behavior of these functions and the behavior of the policy obtained through these functions. We make an attempt on that in the next section.

In terms of transition function, there can be both deterministic and stochastic formulations. There are two types of transitions in the model. First is the type of transition resulting from application of an action, the second is the exogenous transition caused by environmental factors. For example, arrival of a customer or an item being desired by a customer is exogenous event whereas assignment of an item to a robot is caused by supervisor agent. Exogenous transitions are not modeled in the problem. Whereas the action based transitions are modeled via a probability tensor of the form $T(s', a, s)$ that represents the probability of reaching the state s' from state s by executing action a. First we formulate the deterministic transitions for our problem and then we talk about uncertainties.

$$T(s', a_{i,j,k}, s) = \begin{cases} s': d^{i,j}(s') = 0, c^k(s') = c^k(s) + 1, \\ \quad if\ d^{i,j}(s) = 1 \\ \\ s': s' = s, \\ \quad otherwise \end{cases} \quad (7)$$

In the above equation, the action $a_{i,j,k}$ represents the action to assign the i^{th} desired item by j^{th} customer to the k^{th} robot. Notice that the action results in the assignment of the item if the item is actually desired by the customer (i.e. $d^{i,j} = 1$). Otherwise no change is incurred by the action. Notice that there are no probabilities involved in (7) and so it is a deterministic transition function.

While the model of the MDP has the capability to represent stochastic transitions, we confine our discussion in this paper to the deterministic ones. This is to keep the focus on the tradeoff involved in the selection of the appropriate reward function.

4. Calculation of Optimal Policy

Once the problem has been formulated in terms of states, actions, reward function and transition function, the next step is to determine the optimal policy. The advantage of modeling the problem as we have in section II is that there are many algorithms available in the literature for the calculation of the optimal policy. Here we discuss a few prominent ones.

4.1. Value Iteration

The idea of value iteration is to define the value of a state based on how good it is (in terms of the reward function) and how good the states to which we can transition from the given state are. The value of a state is given by

$$V(s) = {}^{max}_{a}(R(s,a) + \gamma \sum_{s'} T(s', a, s)V(s')) \quad (8)$$

Here, $V(s)$ represents the value of the state s and γ is the discount factor. Note that the transitions in our case are all deterministic therefore, for our case, the simplified value iteration equation shall be

$$V(s) = {}^{max}_{a}(R(s,a) + \gamma \sum_{s'} V(s')). \quad (9)$$

In order to calculate the optimal policy, one needs to iterate on the values of the states starting from an initial guess. The initial guess may be $V(s) = {}^{max}_{a} R(s,a), \forall s$. The equation for iteration is

$$V_{t+1}(s) \leftarrow {}^{max}_{a}(R(s,a) + \gamma \sum_{s'} V_t(s')) \quad (10)$$

It is an established result [26] that the iterations in (10) converge to an optimal value with respect to an expected discounted reward given by

$$E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]. \quad (11)$$

Here, E[.] represents the expected value function. In the deterministic case, the expected value is equal to the average value and is represented as

$$\lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n} \gamma^t R(s_t, a_t). \quad (12)$$

Once the iterations in (11) have converged, the optimal policy is calculated using the following equation.

$$\pi^* = \underset{a}{\operatorname{argmax}}(R(s,a) + \sum_{s'} V(s')). \quad (13)$$

4.2. Policy Iteration

An alternative algorithm for the calculation of the optimal policy is policy iteration. The policy iteration algorithm begins with an initial policy π^0 and it consists of two repetitive steps i.e. policy evaluation and policy improvement. The policy evaluation step calculates the value of each state given a fixed policy.

$$V_{t+1}(s) = R(s, \pi(s)) + \sum_{s'} V_t(s'). \quad (14)$$

Here the difference from value iteration is that instead of updating based on maximization over all actions (as in (10)), we update based on the current policy. Next step is policy improvement that is carried out as:

$$\pi_{t+1}(s) = max \left(\begin{pmatrix} \max_a R(s,a) + \sum_{s'} V_t(s') \end{pmatrix}, \begin{pmatrix} R(s, \pi_t(s)) + \sum_{s'} V_t(s') \end{pmatrix} \right) \quad (15)$$

The iterations stop when $\pi_{t+1}(s) = \pi_t(s)$. There is an established result [26] that the policy iteration does converge, and it converges to the policy that is optimal with respect to (11).

4.3. Backward Induction

The two algorithms discussed so far are for the infinite decision-making horizon. There are cases where the decision making is only allowed within limited time frame or in other words, the given tasks have to be completed within finite number of steps. For such cases, the calculation of the optimal policy is carried out using the backward induction algorithm. The optimization criterion for the finite horizon case is

$$\frac{1}{n} \sum_{t=0}^{n} \gamma^t R(s_t, a_t), n < \infty \quad (16)$$

In backward induction, the terminal value of each ($V_n(s)$) state has to be initialized or known a priory. The iterations are performed in reverse order i.e. setting $t = n$

– 1 in the beginning and updating value of each state as

$$V_t(s) \leftarrow \max_a (R(s,a) + \gamma \sum_{s'} V_{t+1}(s')). \quad (17)$$

The value of t is decreased and (18) is recomputed till $t = 0$. After that, the optimal policy is computed as

$$\pi_t{}^* = \operatorname{argmax}_a (R(s_t,a) + \sum_{s'} V_t(s')). \quad (18)$$

Notice that here the optimal policy it dependent on t which means that it is time varying. This is not the case with infinite horizon optimal policy.

4.4. Scalability Issues

Given the state space formulation in (1) and the computations involved in the algorithms discussed above, it is important to analyze the complexity of the problem at hand. Let's evaluate the number of states we have to deal with in case of a simple example. Assume there are 10 aisles in a superstore ($p = 10$), there are five robots ($n = 5$), the store can handle 20 customers at a time ($m = 20$), and each robot can carry four items ($q = 4$). For this simple case, the number of states as per (2) amounts to $p^n \times 2^{pm} \times m^m \times q^n \cong 1.72 \times 10^{94}$. Obviously we cannot hope to solve such a humongous problem (not even the cloud computing can solve it in a human life time). So, the question is, how do we solve this issue? Fortunately, there are many answers available in literature to this question. One simple answer is to decompose the problem into smaller sub-problems. This may result in the loss of optimality in the solution but of course a near optimal solution is much better than no solution.

Before we jump into decomposition right away, it is important to realize which variables are contributing the most to the complexity of the problem. For example, if we reduce the number of customers by half (i.e., $m = 10$), then the resulting state space size becomes nearly 1.29×10^{48} which is significantly lesser than the original size. Further reducing the number of customers say m = 3, number of robots to $n = 2$, and number of aisles to $p = 4$, we get the problem size reduced to about 28 million states. This can be solved by an Intel core i5 laptop in a few minutes. So we divide the number of aisles and the number of customers among different supervisors. Each supervisor has four specific aisles and can handle up to three customers and has two robots under its supervision. In this manner shopping assistant problem for arbitrarily large superstore can be handled.

5. Comparison with Existing Approaches

In this section we present a qualitative comparison of our proposed method with the existing approach. To the best of our knowledge, there is no existing MDP model or any other model for the problem that is exactly the same as setup in this paper. The closest approach is that in [28] where a single robot is considered. However, the approach in [28] does not present explicit form of the transition probability function.

Table 1 presents a qualitative comparison of our proposed approach with three of the existing approaches. Specifically, we have selected three qualities of a task assignment solution, namely, modeling of the uncertainty in the problem, incorporation of multiagent operation, and optimization of the task assignment decisions. It is evident that out proposed approach caters for all of the three qualities. The approach in [23] considers the task assignment for robots in a warehouse setup where a heuristic approach has been employed. This approach does cater for the multiagent setup but is neither optimized (with respect to any specified cost function) nor is there any model of the uncertainty. Similarly, in [25], the optimization has been carried out but the model of the uncertainty is lacking. Finally, in [28], the approach considers a single robot and also the explicit model of the uncertainty has not been discussed.

Table 1. Qualitative Comparison with Existing Approaches.

Approach Reference	Modeling of Uncertainty	Multiagent	Optimized
[23]	No	Yes	No
[25]	No	Yes	Yes
[28]	No	No	Yes
Proposed	Yes	Yes	Yes

6. Simulation Based Analysis

In this section we present the comparative study of the proposed reward functions and discuss the behavior of the resulting optimal policy where applicable. In the proceeding analysis, we have adopted the example with the following parameter values.

Table 2. Parameter Values.

Parameter	**Value**
Number of customers at one time (m)	3
Number of available robots (n)	2
Number of aisles (p)	4
Maximum number of items that a robot can carry (q)	4
Reward weightage (α_1)	10 (is 0.1 for exponential reward only)
Reward weightage (α_2)	10 (is 0.1 for exponential reward only)
Number of states (z)	28,311,552

In order to demonstrate the comparison between

reward functions in (3)-(6), we select a specific state where all of the three customers desire all four items. Furthermore, one of the robots is in aisle 1 and the second robot is in aisle 4. Each robot is assumed to be initially carrying one item. Customer priority value is set by the index number i.e. customer 1 has lowest priority value and customer 3 has highest priority value. The state we selected is given as

$$\begin{aligned} &s = \{L, D, C, O\}, \\ &L = \{1,2,3\}, \\ &D = \{1,1,1,1,1,1,1,1,1,1,1,1\}, \\ &C = \{1,1\}, \\ &O = \{1,4\}. \end{aligned} \tag{19}$$

As indicated by (3) there are twenty-four actions. For clarity, the description of each action is given in Table 2. Fig. 1 show the values of the reward that we obtain against each of the 24 actions at state (19). As per the results, actions 1, 8, 9, 16, 17, and 24 have highest values. This is reasonable since all these actions correspond to either assigning robot 1 an item in the aisle 1 or assigning robot 2 an item in the aisle 4. Recall that in our case study, robot 1 is in aisle 1 and robot 2 is in aisle 4. Note that although the values of the rewards vary depending upon what reward function is used but the basic shape of the curve is the same for all four reward functions. This is not obvious as we know that for any function f, the shape of $log(f)$ and $exp(f)$ do not match with each other or with shape of f in general. From the comparison in Fig. 1, one may be tempted to infer that all four reward functions yield identical optimal policy. While this is true for our example and only at state (19), a generalized result requires rigorous mathematical analysis and cannot be based on specific simulation results. The results however do help in pointing out the direction of the mathematical analysis that is something left to be done as future work.

An interesting property to extract from Fig. 1 is the ratio of the maximum and minimum value of the reward function corresponding to various actions. A high ratio is desirable since it separates out the best action from the worst one. The calculated ratios are provided in Table 3. It is shown that the product reward function from Eq. (6) provides the best maximum to minimum reward ratio of 4.00. Second best is the logarithmic reward function with ratio of 1.67.

Table 3. Description of the actions with respect to their indices.

Action Index	Description
1	Assign item 1 desired by customer 1 to robot 1
2	Assign item 1 desired by customer 1 to robot 2
3	Assign item 2 desired by customer 1 to robot 1
4	Assign item 2 desired by customer 1 to robot 2
5	Assign item 3 desired by customer 1 to robot 1
6	Assign item 3 desired by customer 1 to robot 2
7	Assign item 4 desired by customer 1 to robot 1
8	Assign item 4 desired by customer 1 to robot 2
9	Assign item 1 desired by customer 2 to robot 1
10	Assign item 1 desired by customer 2 to robot 2
11	Assign item 2 desired by customer 2 to robot 1
12	Assign item 2 desired by customer 2 to robot 2
13	Assign item 3 desired by customer 2 to robot 1
14	Assign item 3 desired by customer 2 to robot 2
15	Assign item 4 desired by customer 2 to robot 1
16	Assign item 4 desired by customer 2 to robot 2
17	Assign item 1 desired by customer 3 to robot 1
18	Assign item 1 desired by customer 3 to robot 2
19	Assign item 2 desired by customer 3 to robot 1
20	Assign item 2 desired by customer 3 to robot 2
21	Assign item 3 desired by customer 3 to robot 1
22	Assign item 3 desired by customer 3 to robot 2
23	Assign item 4 desired by customer 3 to robot 1
24	Assign item 4 desired by customer 3 to robot 2

Table 4. Maximum to minimum ratios of the reward values.

Reward Function Type	$max(R)/min(R)$
Linear (Eq. (3))	1.0353
Exponential (Eq. (4))	1.0012
Logarithmic (Eq. (5))	1.6705
Product (Eq. (6))	4.00

Table 5. Variations in *D*.

Index	Values in D
1	{1,0,0,0,0,0,0,0,0,0,0,0}
2	{1,1,0,0,0,0,0,0,0,0,0,0}
3	{1,1,1,0,0,0,0,0,0,0,0,0}
4	{1,1,1,1,0,0,0,0,0,0,0,0}
5	{1,1,1,1,1,0,0,0,0,0,0,0}
6	{1,1,1,1,1,1,0,0,0,0,0,0}
7	{1,1,1,1,1,1,1,0,0,0,0,0}
8	{1,1,1,1,1,1,1,1,0,0,0,0}
9	{1,1,1,1,1,1,1,1,1,0,0,0}
10	{1,1,1,1,1,1,1,1,1,1,0,0}
11	{1,1,1,1,1,1,1,1,1,1,1,0}
12	{1,1,1,1,1,1,1,1,1,1,1,1}

DOI:10.4186/ej.2021.25.11.1

Our second analysis is the study of the variation in the value of reward for a particular action as the state changes. Specifically, we study the change in the desirability of various items by various customers. The state variables are the same as in (19) except for the variable D. The evolution of the variable D is given in Table 4.

The resulting values for the best and worst actions are shown (Fig. 2) as a function of the variations in D. Note that there are multiple best and worst actions in Fig. 1 (here we selected action 1 out of the best actions and action 2 out of the worst actions). The values of the actions decrease as the number of desirable items is increased. This shows that an item being desirable (and not assigned to one of the robots) is discouraged by the reward function. The change in the ratio of the action values is indicated in Table 5. The results show that the product function holds the ratio better than the other three functions.

Table 6. Variation in the max/min ratio of the action values in response to the variations in D.

Reward Function Type	$max(R_{start})/min(R_{start})$	$max(R_{end})/min(R_{end})$
Linear (Eq. (3))	1.0278	1.0353
Exponential (Eq. (4))	1.0001	1.0012
Logarithmic (Eq. (5))	1.6588	1.6705
Product (Eq. (6))	4.00	4.00

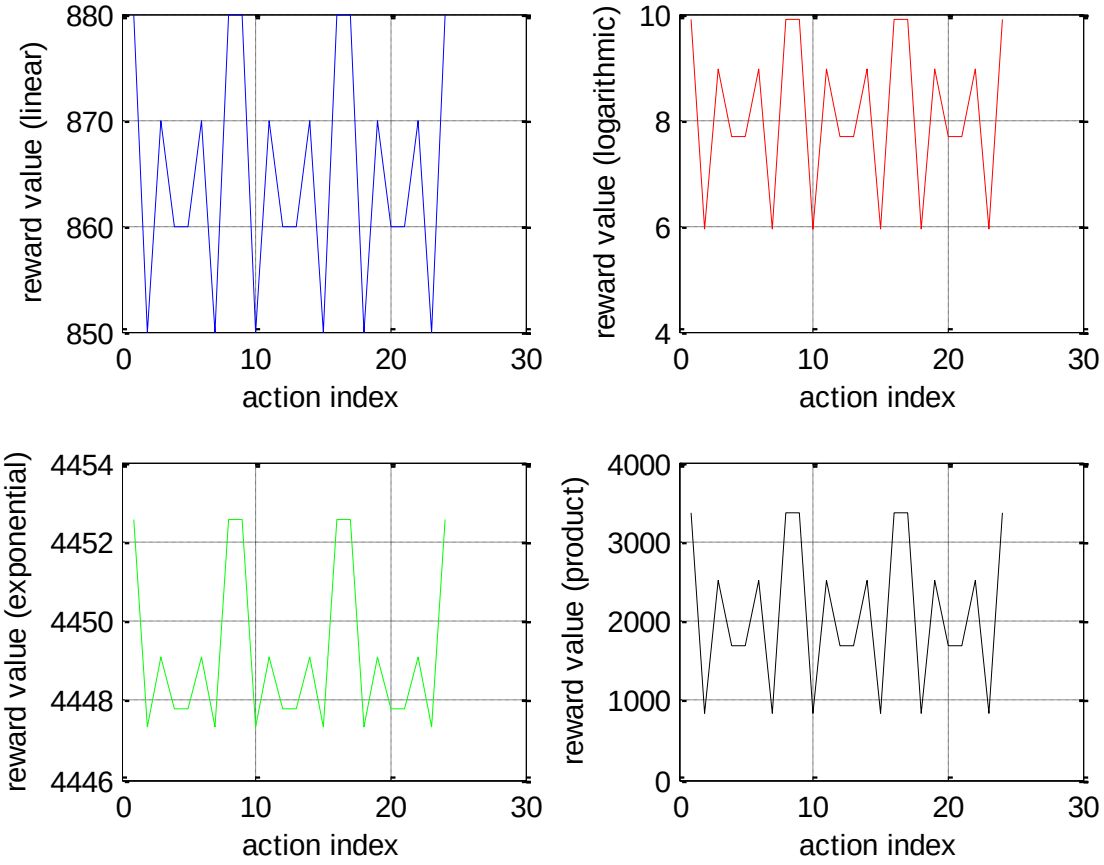

Fig. 2. Values of various actions from a specific state.

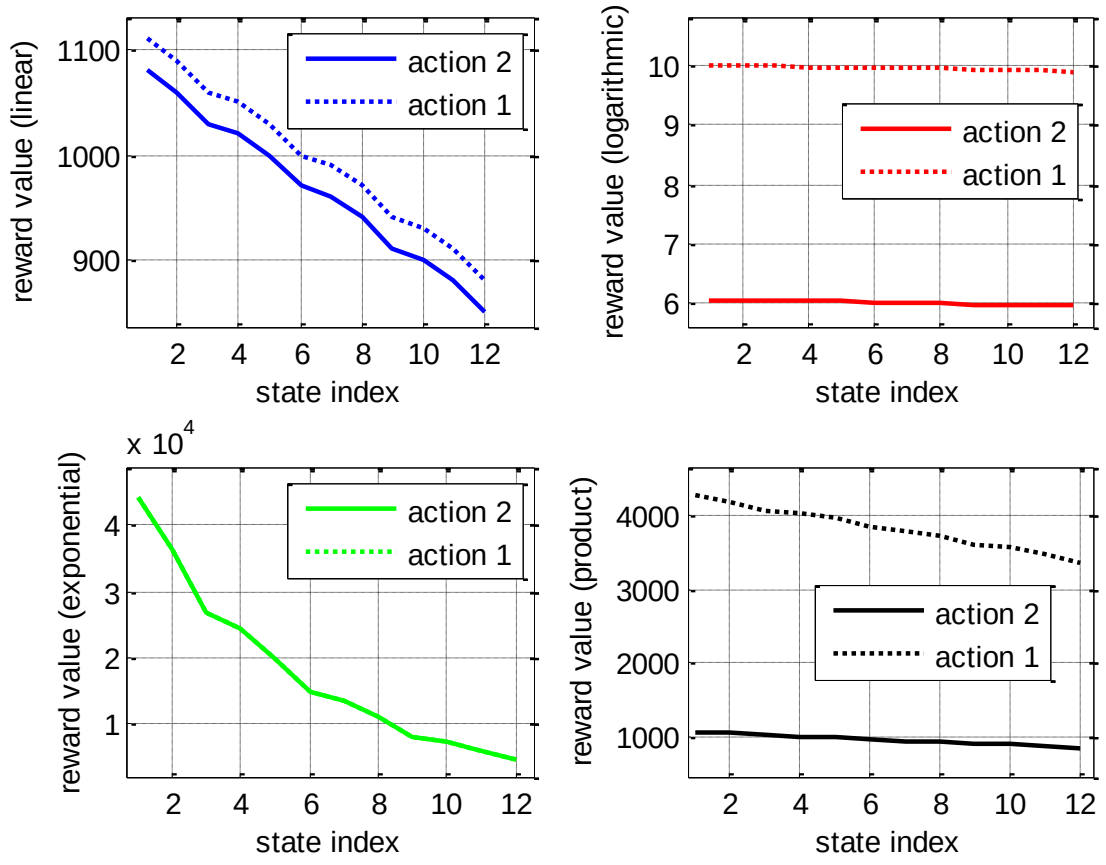

Fig. 3. Variations in the values of actions in response to the variations in *D*.

7. Conclusions

An MDP model related to the task assignment problem for the fetching robots in a large superstore has been proposed. The proposed model incorporates the uncertainty involved in the desirability of the items to be fetched and in the arrival rate of the customers. The reward function of the proposed MDP has been analyzed in terms of its impact on making task assignment decisions. The proposed approach does not incorporate the path planning or the low-level reference tracking control of the robots.

Scalability issues related with the proposed model have been discussed and it has been pointed out that for larger problems, the decomposition of MDP is the most feasible approach. Comparison of the reward function in the results section indicates that the logarithmic equation of the reward results in maximum discrimination among the good and bad decisions. Comparison with the existing approaches indicates that the proposed approach is superior in terms of formally modeling the uncertainty involved in the problem.

References

[1] H.-M. Gross, et al., "TOOMAS: Interactive shopping guide robots in everyday use-final implementation and experiences from long-term field trials," *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009.

[2] V. Kulyukin, C. Gharpure, and J. Nicholson, "Robocart: Toward robot-assisted navigation of grocery stores by the visually impaired," in *Intelligent Robots and Systems, 2005, (IROS 2005), IEEE/RSJ International Conference on,* IEEE.

[3] N. Doering, et al., "User-centered design and evaluation of a mobile shopping robot," *International Journal of Social Robotics*, vol. 7, no. 2, pp. 203-225, 2015.

[4] H.-M. Gross, et al., "Shopbot: Progress in developing an interactive mobile shopping assistant for everyday use," *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*. IEEE, 2008.

[5] G. Phanomchoeng, M. Saadi, P. Sasithong, J. Tangmongkhonsuk, S. K. Wijayasekara, and L. Wuttisittikulkij, "Hardware software co-design of a

farming robot," *Engineering Journal*, vol. 24, no. 1, pp. 199-208, 2020.

[6] M. Hersh, "Overcoming barriers and increasing independence-service robots for elderly and disabled people," *International Journal of Advanced Robotic Systems*, vol. 12, 2015.

[7] H.-M. Gross, et al., "Vision-based Monte Carlo self-localization for a mobile service robot acting as shopping assistant in a home store," *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, IEEE, 2002, vol. 1.

[8] T. Tomizawa, A. Ohya, and S. Yuta, "Remote shopping robot system—Development of a hand mechanism for grasping fresh foods in a supermarket," *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006.

[9] T. Tomizawa, et al., "Remote food shopping robot system in a supermarket-realization of the shopping task from remote places," *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*. IEEE, 2007.

[10] Y. L. Ng, et al., "Automatic human guided shopping trolley with smart shopping system," *Jurnal Teknologi*, vol. 73, no. 3, 2015.

[11] L. Iocchi, et al., "Personalized short-term multi-modal interaction for social robots assisting users in shopping malls," *Social Robotics*, pp. 264-274, 2015.

[12] Y. Iwamura, et al., "Do elderly people prefer a conversational humanoid as a shopping assistant partner in supermarkets?," *Human-Robot Interaction (HRI), 2011 6th ACM/IEEE International Conference on*, IEEE, 2011.

[13] A. A. M. Teodoro, O. S. M. Gomes, M. Saadi, B. A. Silva, R. L. Rosa, and D. Z. Rodríguez, "An FPGA-based performance evaluation of artificial neural network architecture algorithm for IoT," *Wireless Personal Communications*, pp. 1-32, 2021.

[14] T. Kanda, et al., "An affective guide robot in a shopping mall," in *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*, ACM, 2009.

[15] O. B. Kwon and N. Sadeh, "Applying case-based reasoning and multi-agent intelligent system to context-aware comparative shopping," *Decision Support Systems*, vol. 37, no. 2, pp. 199-213, 2004.

[16] F. Menczer, A. E. Monge, and W. N. Street, "Adaptive assistants for customized e-shopping," *IEEE Intelligent Systems*, vol. 6, pp. 12-19, 2002.

[17] C. L. R. McGhan, A. Nasir, and E. M. Atkins, "Human intent prediction using Markov decision processes," *Journal of Aerospace Information Systems*, vol. 12, no. 5, pp. 393-397, May 2015, doi: 10.2514/1.I010090

[18] B. Chandrasekaran and J. M. Conrad, "Human-robot collaboration: A survey," in *SoutheastCon 2015*, IEEE, 2015.

[19] K. Thongsanit, R. Boondisakulchok, and W. Tharmmaphornphilas, "Heuristic for task-worker assignment with varying learning slopes," *Eng. J.*, vol. 14, no. 2, pp. 1-14, Mar. 2010.

[20] S. Swangnop and P. Chaovalitwongse, "Joint requirement of two multi-skill resource types in multi-period multi-site assignment problem," *Eng. J.*, vol. 19, no. 1, pp. 51-65, Jan. 2015.

[21] J. C. Silva, M. Saadi, L. Wuttisittikulkij, D. R. Militani, R. L. Rosa, D. Z. Rodríguez, and S. Al Otaibi, "Light-field imaging reconstruction using deep learning enabling intelligent autonomous transportation system," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[22] N. Phumchusri and P. Kitpipit, "Warehouse layout design for an automotive raw material supplier," *Eng. J.*, vol. 21, no. 7, pp. 361-387, Dec. 2017.

[23] A. Bolu and Ö. Korçak, "Adaptive task planning for multi-robot smart warehouse," *IEEE Access*, vol. 9 pp. 27346-27358, 202.

[24] R. Sirovetnukul and P. Chutima, "The impact of walking time on u-shaped assembly line worker allocation problems," *Eng. J.*, vol. 14, no. 2, pp. 53-78, Apr. 2010.

[25] A. Parnianifard, M. Saadi, M. Pengnoo, M. A. Imran, S. Al Otaibi, P. Sasithong, P. Vanichchanunt, T. Polysuwan, and L. Wuttisittikulkij, "Hybrid metamodeling/metaheuristic assisted multi-transmitters placement planning," *CMC-Computers Materials & Continua*, vol. 68, no. 1, pp. 569-587, 2021.

[26] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[27] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach,* 2nd ed. Upper Saddle River, New Jersey: Prentice-Hall, 2005.

[28] R. Gillani and A. Nasir, "Incorporating artificial intelligence in shopping assistance robot using Markov Decision Process," *2016 International Conference on Intelligent Systems Engineering (ICISE)*, Islamabad, Pakistan, 2016, pp. 94-99.

Ali Nasir, photograph and biography not available at the time of publication.

Muhammad Saadi, photograph and biography not available at the time of publication.

Rida Gelani, photograph and biography not available at the time of publication.

Faisal Mustafa, photograph and biography not available at the time of publication.