RAIRO-Oper. Res. 53 (2019) 1749–1761
https://doi.org/10.1051/ro/2018081

RAIRO Operations Research
www.rairo-ro.org

A LINEAR PROGRAMMING BASED APPROACH FOR COMPOSITE-ACTION MARKOV DECISION PROCESSES

Zhicong Zhang[1,*], Shuai Li[1], Xiaohui Yan[1] and Liangwei Zhang[1]

Abstract. We study a time homogeneous discrete composite-action Markov decision process (CMDP) which needs to make multiple decisions at each state. In this particular Markov decision process, the state variables are divided into two separable sets and a two-dimensional composite action is chosen at each decision epoch. To solve a composite-action Markov decision process, we propose a novel linear programming model (Contracted Linear Programming Model, CLPM). We show that the CLPM model obtains the optimal state values of a CMDP process. We analyze and compare the number of variables and constraints of the CLPM model and the Traditional Linear Programming Model (TLPM). Computational experiments compare running times and memory usage of the two models. The CLPM model outperforms the TLPM model in both time complexity and space complexity by theoretical analysis and computational experiments.

Mathematics Subject Classification. 90C40, 90C05.

Received December 12, 2017. Accepted September 19, 2018.

1. Introduction

Markov decision processes (MDP) with multi-dimensional action space or multi-actors are more complicated than those with one-dimensional action space. In the literature, some researchers apply this kind of Markov decision processes to modeling two-stage tandem queueing systems. Kim and Dudin [6] studied a two-stage multi-server tandem queue with two types of processed customers. The queueing system's behavior is described in terms of the multi-dimensional asymptotically quasi-Toeplitz continuous time Markov chain. Ahn *et al.* [1] considered an optimal control problem of two parallel servers in a two-stage tandem queuing system with two flexible servers. The problem involves two actors and two actions. They showed that in states where both workers can be assigned to either station, assigning both to the same station is optimal and when servers can collaborate, they always work at the same station. Kaufman *et al.* [5] also considered a two-station tandem queueing system where customers arrive according to a Poisson process and must receive service at both stations before leaving the system. They analyzed three scenarios for the fluctuations of workforce level. They showed in each scenario that all workers should be allocated to one queue or the other (never split between queues) and that they should serve exhaustively at one of the queues depending on the direction of an inequality.

Keywords. Markov decision process, linear programming, optimal state value, action.

[1] Department of Industrial Engineering, Dongguan University of Technology, 523808 Dongguan, PR China.
[*]Corresponding author: stephen1998@gmail.com

They also showed in the unrestricted case that the optimal number of workers to have in the system is non-decreasing in the number of customers in either queue. Wu *et al.* [13] studied the allocation of reconfigurable resources in a serial line with machine failures. They showed in the two-station case that transition monotone optimal policies exist and discussed heuristics based on the two-server model that reduces average holding costs significantly. Pandelis [9] considered two-stage tandem queuing systems with dedicated servers in each station and flexible collaborating servers that can serve in both stations. He assumed exponential service times, linear holding costs, and operating costs incurred by the servers at rates proportional to their speeds, and showed that the optimal allocation of flexible servers is determined by a transition-monotone policy. Pandelis [10] analyzed a model containing two interconnected queues attended by two specialized and one flexible server with time varying service rates. Assuming exponential processing times and linear holding costs, he derives properties of server allocation policies that minimize expected costs over an infinite time horizon. Other research on tandem queueing systems with Markov decision process models includes Andradóttir and Ayhan [3], Kırkızlar *et al.* [7], Hasenbein & Kim [4] and Kırkızlar *et al.* [8].

To reveal the insights of multi-actors Markov decision processes, Ahn and Righter [2] gave a general reformulation of multi-actor Markov decision processes and showed that there is a tendency for the actors to take the same action whenever possible. They made two key assumptions. The first one is that firing rates are multiplicative so that some actors are uniformly faster or slower than the others. The second one is that state transitions depend on the action chosen and not on which actor chooses the action. This study reduces the complexity of the proposed problem, either facilitating numerical computation of the optimal policy or providing a basis for a heuristic. By extending the previous work, Pandelis [11] studied controlled Markov processes where multiple decisions need to be made for each state. He included state dependent costs, actor operating costs, state transitions depending on the actor as well as on the action chosen, arbitrary sets of admissible actions for each actor, and restrictions on the number of actors that can choose the same action. Under the basic assumption that each actor's speed is the same for all of his admissible actions, he identified models where the optimal actions are contained in some subset of the action space. The results reduced the computational complexity of the search for an optimal policy.

We study a particular Markov decision process, composite-action Markov decision process (CMDP), which selects a two-dimensional action at a decision epoch, *i.e.* two decisions need to be made for each state. Take the example of a multi-mode single-server queuing system with multi-type jobs to illustrate a CMDP process. The server in this queuing system needs to serve multiple kinds of jobs who dynamically arrive and wait in the queue if the server is busy. The server has several working modes. Different working modes correspond to different service cost and different service efficiency. When the server finishes serving a job, it must choose a two-dimensional action: the first sub-action switches the server's working mode to an appropriate one, and then the second sub-action selects a job waiting in the queue to serve. Switching the server's working mode needs a specific conversion cost. The values of service cost and service time for the selected job relies on the current working mode. The controlling of a multi-mode single-server queuing system with multi-type jobs can be converted into a CMDP model. We propose a novel linear programming based approach to solve a CMDP process. We prove the optimality of the proposed linear programming based approach and show that this approach is superior to the traditional approach by theoretical analysis and numerical experiments.

This paper is organized as follows: a CMDP process is formulated in Section 2, the Contracted Linear Programming Model (CLPM) is proposed in Section 3, the property of the CLPM model for a CMDP process is proved and the scale of the CLPM model is analyzed in Section 4, computational experiments are conducted in Section 5 and conclusions are drawn in Section 6.

2. The model of composite-action Markov decision process

2.1. Formulation of composite-action Markov decision process

Composite-action Markov decision process (CMDP) is formulated as follows. A CMDP process is a particular time homogeneous discrete Markov decision process with finite state space S and value function $f\colon S \to R$, where

R denotes a set of real numbers. This MDP is observed at time points $t \in T$, where $T = \{0, 1, 2, \ldots\}$. At some time point $t \in T$, let s denote the state of the process. s is a state vector represented as $s = (i_{(1)}, i_{(2)})$, where $i_{(1)}(i_{(1)} \in S_1)$ and $i_{(2)}(i_{(2)} \in S_2)$ are two complementary sub-state vectors of variables, *i.e.* all state variables are divided into two subsets $i_{(1)}$ and $i_{(2)}$. S_1 is the spanned state space by the state variables in sub-state vector $i_{(1)}$ and S_2 is the spanned state space by the state variables in sub-state vector $i_{(2)}$. Hence, $S = S_1 \times S_2$. $i_{(1)}$ and $i_{(2)}$ are also called the first sub-state vector and the second sub-state vector respectively. A CMDP process is a Markov decision process with composite actions. After observing the state of the process at a decision epoch, a composite action $a = (a_{(1)}, a_{(2)})$ containing two sequential sub-actions $a_{(1)}$ and $a_{(2)}$ must be chosen. Taking sub-action $a_{(1)}$ only changes the state variables in $i_{(1)}$ and it does not change the state variables in $i_{(2)}$ directly. Similarly, taking sub-action $a_{(2)}$ only changes the state variables in $i_{(2)}$ and it does not change the state variables in $i_{(1)}$. Let $A_1(i_{(1)})$ denote the set of possible sub-actions when the first sub-state vector is $i_{(1)}$. That is, if the first sub-state vector of state variables at a decision epoch is $i_{(1)}$, and the two sequential sub-actions $a_{(1)}$ and $a_{(2)}$ are chosen at this state, then $a_{(1)} \in A_1(i_{(1)})$. Similarly, $A_2(i_{(2)})$ denotes the set of possible sub-actions when the second sub-state vector is $i_{(2)}$. That is, if the second sub-state vector of state variables at a decision epoch is $i_{(2)}$, and the two sequential sub-actions $a_{(1)}$ and $a_{(2)}$ are chosen at this state, then $a_{(2)} \in A_2(i_{(2)})$. For convenience, let $A(i_{(1)}, i_{(2)})$ denote the set of possible composite actions at state $(i_{(1)}, i_{(2)})$.

After sequentially choosing sub-actions $a_{(1)}$ and $a_{(2)}$, they are implemented and then a state transition takes place. The state transition of $i_{(1)}$, only caused by sub-action $a_{(1)}$, is deterministic and independent of sub-action $a_{(2)}$. Let H_1 denote the transition operator of the first sub-state vector. Thus, the equation $H_1(i_{(1)}, a_{(1)}) = j_{(1)}$ means that the first sub-state vector transfers from $i_{(1)}$ into $j_{(1)}$ deterministically by taking sub-action $a_{(1)}$. However, the state transition of sub-state vector $i_{(2)}$ depends on the second sub-action $a_{(2)}$, and this transition is stochastic. Suppose a state transition from state $(i_{(1)}, i_{(2)})$ to state $(j_{(1)}, j_{(2)})$ takes place by taking action $(a_{(1)}, a_{(2)})$. This state transition can also be regarded as containing two sub-transitions, one is from state $(i_{(1)}, i_{(2)})$ to the interim state $(j_{(1)}, i_{(2)})$ caused by $a_{(1)}$, the other is from the interim state $(j_{(1)}, i_{(2)})$ to state $(j_{(1)}, j_{(2)})$ caused by $a_{(2)}$. Trivially, there exists a special null action $a^0_{(1)}(a^0_{(1)} \in A_1(i_{(1)}))$ for any $i_{(1)} \in S_1$ by taking which the values of state variables in $i_{(1)}$ do not change, *i.e.* $H_1(i_{(1)}, a^0_{(1)}) = i_{(1)}$ holds for any $i_{(1)} \in S_1$. Let $P(j_{(1)}, j_{(2)}|i_{(1)}, i_{(2)}; a_{(1)}, a_{(2)})$ denote the probability of state transition from $(i_{(1)}, i_{(2)})$ to $(j_{(1)}, j_{(2)})$ by taking composite action $(a_{(1)}, a_{(2)})$, where $j_{(1)} = H_1(i_{(1)}, a_{(1)})$. For any $i_{(1)} \in S_1$, $a_{(1)} \in A_1(i_{(1)})$ and $a_{(2)} \in A_2(i_{(2)})$, we have $\sum_{j_{(2)} \in S_2} P(j_{(1)}, j_{(2)}|i_{(1)}, i_{(2)}; a_{(1)}, a_{(2)}) = 1$. Trivially, for any $i_{(1)} \in S_1$, $i_{(2)} \in S_2$ and $a_{(2)} \in A_2(i_{(2)})$, we have $\sum_{j_{(2)} \in S_2} P(i_{(1)}, j_{(2)}|i_{(1)}, i_{(2)}; a^0_{(1)}, a_{(2)}) = 1$. Let $P(i_{(1)}, j_{(2)}|i_{(1)}, i_{(2)}; a_{(2)})$ denote the probability of state transition from $(i_{(1)}, i_{(2)})$ to $(i_{(1)}, j_{(2)})$ by taking sub-action $a_{(2)}$. Since the state transition of $i_{(1)}$ by taking sub-action $a_{(1)}$ is deterministic and independent of $a_{(2)}$, it follows that

$$P(j_{(1)}, j_{(2)}|j_{(1)}, i_{(2)}; a_{(2)}) = P(j_{(1)}, j_{(2)}|i_{(1)}, i_{(2)}; a_{(1)}, a_{(2)}), \tag{2.1}$$

where $j_{(1)} = H_1(i_{(1)}, a_{(1)})$.

Taking composite action $(a_{(1)}, a_{(2)})$ in state $(i_{(1)}, i_{(2)})$ obtains a separable reward denoted by $r(i_{(1)}, i_{(2)}; a_{(1)}, a_{(2)})$. Reward $r(i_{(1)}, i_{(2)}; a_{(1)}, a_{(2)})$ is the sum of $r_1(i_{(1)}, i_{(2)}; a_{(1)})$ and $r_2(j_{(1)}, i_{(2)}; a_{(2)})$, where $j_{(1)} = H_1(i_{(1)}, a_{(1)})$. $r_1(i_{(1)}, i_{(2)}; a_{(1)})$ denotes the reward received by taking sub-action $a_{(1)}$ at state $(i_{(1)}, i_{(2)})$ and its value is determined by both $(i_{(1)}, i_{(2)})$ and $a_{(1)}$. $r_2(j_{(1)}, i_{(2)}; a_{(2)})$ denotes the reward received by taking sub-action $a_{(2)}$ at state $(j_{(1)}, i_{(2)})$. The value of $r_2(j_{(1)}, i_{(2)}; a_{(2)})$ relies on $j_{(1)}$, $i_{(2)}$ and $a_{(2)}$. Because $j_{(1)}$ depends on $i_{(1)}$ and $a_{(1)}$, the value of $r_2(j_{(1)}, i_{(2)}; a_{(2)})$ is essentially determined by $i_{(1)}$, $i_{(2)}$, $a_{(1)}$ and $a_{(2)}$. Assume that all the reward functions are bounded.

For any policy π, we define

$$V_\pi(i_{(1)}, i_{(2)}) = E_\pi[\sum_{t=0}^{+\infty} \beta^t r(s_t; a^t_{(1)}, a^t_{(2)})|s_0 = (i_{(1)}, i_{(2)})], \quad (i_{(1)}, i_{(2)}) \in S \tag{2.2}$$

where E_π represents the conditional expectation given that the policy π is employed, $\beta(0<\beta<1)$ is the discount factor, $s_t = (i^t_{(1)}, i^t_{(2)})$ represents the state at decision epoch t, and $(a^t_{(1)}, a^t_{(2)})$ represents the composite action

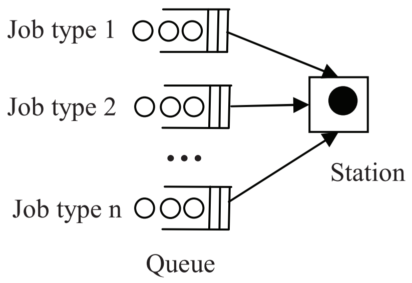

FIGURE 1. A multi-mode station queueing system with multi-type jobs.

taken following policy π at decision epoch t. Thus, $V_\pi(i_{(1)}, i_{(2)})$ represents the expected total discounted reward received in the infinite horizon when the policy π is employed and the initial state is $(i_{(1)}, i_{(2)})$. Define the optimal value function V_β as

$$V_\beta(i_{(1)}, i_{(2)}) = \sup_{\pi\in\Pi} V_\pi(i_{(1)}, i_{(2)}), \quad (i_{(1)}, i_{(2)}) \in S \tag{2.3}$$

where Π represents the policy space. The policy space Π contains the policies which choose a composite action $(a^t_{(1)}, a^t_{(2)})$ containing two sequential sub-actions at state s_t, where taking sub-action $a^t_{(1)}$ only changes the sub-state vector $i^t_{(1)}$ and taking sub-action $a^t_{(2)}$ only changes the sub-state vector $i^t_{(2)}$. V_β is a bounded function defined on the state space. A policy $\pi*$ is said to be optimal if for all $(i_{(1)}, i_{(2)}) \in S$,

$$V_{\pi*}(i_{(1)}, i_{(2)}) = V_\beta(i_{(1)}, i_{(2)}). \tag{2.4}$$

Hence, a policy is optimal if its expected total β-discounted reward is maximal for every initial state.

The objective of the decision maker is to find the optimal policy $\pi*$. In summary, composite-action Markov decision process is a special Markov decision process with separable state space, reward representation and state transition by taking composite actions.

2.2. An illustrative example of CMDP processes

In this section, we take a control problem in a single-station queueing system as an illustrative example of CMDP process. The queueing system control problem is described as follows. As shown in Figure 1, there are n types of jobs involved in this queueing system and let J denote the set of job types. Any type of jobs arrives at the queue of the station following Poisson process. The arriving jobs wait in the queue for processing in the station. The queue capacity is limited. For each job type, at most K jobs wait in the queue for the sake of capacity limit. That is, if a job arrives and K jobs of the same type are waiting in the queue, then the arrival job will not enter the queue and thus it is lost. The station has m working modes. The processing time for each job is one unit time, which is independent of the current working mode of the station. However, the processing cost is dependent of the current working mode. That is, for each type of jobs, the station in different working modes corresponds to different processing costs. Consequently, to obtain more revenue, the station needs to switch its working mode to an appropriate one before choosing a job from the queue to process. The mode switch cost is dependent of both the mode before switching and the mode after switching.

This queueing system is a multi-mode station queueing system with multi-type jobs. The state of the queueing system can be represented by vector s, which is composed of state variables and defined as

$$s = [d, Q_1, Q_2, \ldots, Q_n],$$

where d denotes the current working mode of the station and $Q_k (1 \le k \le n)$ denotes the number of jobs belonging to job type k and waiting in the queue. Because at most K jobs of the same job type wait in the queue, we have $0 \leq Q_k \leq K \quad (1 \leq k \leq n)$. The state vector $s(s \in S)$ can be divided into two complementary sub-vectors: $i_{(1)} = d$ and $i_{(2)} = [Q_1, Q_2, \dots, Q_n]$. Let S_1 denote the set of working modes, then we have $i_{(1)} \in S_1$. Let S_2 denote the set of combinations of possible values of the state variables in sub-state vector $i_{(2)}$. That is, $S_2 = \{[Q_1, Q_2, \dots, Q_n] | 0 \leq Q_k \leq K (1 \leq k \leq n)\}$, then we have $i_{(2)} \in S_2$. Hence, the system state space is $S = S_1 \times S_2$.

Suppose when the station completes processing a job, the queueing system is at the tth decision state denoted s_t, where $s_t = [i_{(1)}, i_{(2)}] = [d, Q_1, Q_2, \dots, Q_n]$. At this decision epoch, the station needs to select a composite action composed of two sequential sub-actions: one is to switch the station's mode and the other one is to select a job waiting in the queue to process. The station first selects the first sub-action, denoted $a^t_{(1)}$, which switches the station's mode. $a^t_{(1)}$ is only dependant on $i_{(1)}$ and it is independent of $i_{(2)}$. Let $A_1(i_{(1)})$ denote the set of feasible sub-actions while the station is in mode $i_{(1)}$. For example, $a^t_{(1)} = 3$ means that the station switches its working mode to the third mode. Suppose $a^t_{(1)}$ means switching the working mode to $j_{(1)}$. When the station takes sub-action $a^t_{(1)}$, its working mode immediately and deterministically transfers from mode $i_{(1)}$ to mode $j_{(1)}$ and the station receives a reward $r_{1,t}$, which may be a negative real number. The value of reward $r_{1,t}$ is only dependant on $i_{(1)}$ and $a^t_{(1)}$, and it is independent of $i_{(2)}$. After switching its working mode, if at least one job is waiting in the queue, the station then needs to select the second sub-action $a^t_{(2)}$, which selects a job from the queue to process. After one unit of time, the selected job is completely processed and the system state stochastically transfers to the next state, *i.e.* the $(t+1)$th decision state $s_{t+1} = [j_{(1)}, j_{(2)}]$. The transition probability from state s_t to s_{t+1} is $P(i_{(1)}, j_{(2)} | i_{(1)}, i_{(2)}; a^t_{(2)})$, which is related to $i_{(2)}$, $j_{(2)}$ and $a^t_{(2)}$. When this state transition takes place, the station receives another reward $r_{2,t}$. The value of reward $r_{2,t}$ is due to $j_{(1)}$, $i_{(2)}$ and $a^t_{(2)}$. When the system state transfers to s_{t+1}, the station needs to select the next composite action.

The control objective function of the queueing system is to maximize the discounted reward defined as

$$\max \sum_{t=0}^{+\infty} \beta^t (r_{1,t} + r_{2,t}), \tag{2.5}$$

where β is the discounted factor, $r_{1,t}$ and $r_{2,t}$ denote the two rewards respectively caused by the two sub-actions $a^t_{(1)}$ and $a^t_{(2)}$ at decision state s_t. Thus, the above decision process associated with (s, t) is a discounted composite-action Markov decision process.

3. Two linear programming models for CMDP processes

Value function methods and policy iteration methods are usually adopted to solve discounted Markov decision processes. Some value function methods uses a linear programming model to compute the optimal state values of a Markov decision process and then determine the optimal control policy following

$$a^*(s) = \arg\max_{a \in A(s)} \{r(s, a) + \beta \sum_{j \in S} P(j|s, a) V_\beta(s)\}, \tag{3.1}$$

where $a^*(s)$ denotes the optimal action at state s, $A(s)$ denotes the set of available actions at state s, $r(s, a)$ denotes the reward when action a is taken at state s, S denotes the state space, $P(j|s, a)$ denotes the probability of state transition from state s to state j by taking action a, $V_\beta(s)$ denotes the optimal state value of state s. A discounted CMDP process is a special discounted Markov decision process, so it also can be solved by a linear programming based method. By the approach of constructing the linear programming model for calculating the optimal state values of a traditional Markov decision process, we construct the following Traditional Linear Programming Model (TLPM) for computing the optimal state values of CMDP processes.

- *Traditional Linear Programming Model (TLPM)*:

$$\begin{aligned}
&\min \sum_{(i_{(1)},i_{(2)})\in S} V(i_{(1)},i_{(2)})\\
&\text{s.t. } V(i_{(1)},i_{(2)}) \geq r_1(i_{(1)},i_{(2)};a_{(1)}) + r_2(j_{(1)},i_{(2)};a_{(2)})\\
&\qquad + \beta \sum_{j_{(2)}\in S_2} P(j_{(1)},j_{(2)}|i_{(1)},i_{(2)};a_{(1)},a_{(2)})V(j_{(1)},j_{(2)}),\\
&\forall (i_{(1)},i_{(2)}) \in S, a_{(1)} \in A_1(i_{(1)}),\\
&\quad a_{(2)} \in A_2(i_{(2)}), j_{(1)} = H_1(i_{(1)},a_{(1)})
\end{aligned} \tag{3.2}$$

According to equation (2.1), constraint (3.2) is reformulated as

$$\begin{aligned}
&V(i_{(1)},i_{(2)}) \geq r_1(i_{(1)},i_{(2)};a_{(1)}) + r_2(j_{(1)},i_{(2)};a_{(2)}) + \beta \sum_{j_{(2)}\in S_2} P(j_{(1)},j_{(2)}|j_{(1)},i_{(2)};a_{(2)})V(j_{(1)},j_{(2)}),\\
&\forall (i_{(1)},i_{(2)}) \in S, a_{(1)} \in A_1(i_{(1)}), \quad a_{(2)} \in A_2(i_{(2)})
\end{aligned} \tag{3.3}$$

The optimal solution to the TLPM model is the optimal state value function $V_\beta(i_{(1)},i_{(2)}) \quad ((i_{(1)},i_{(2)}) \in S)$ of a CMDP process. We propose another linear programming model, called the Contracted Linear Programming Model (CLPM), for computing the optimal state values of a CMDP process. The CLPM model is formulated as follows.

- *Contracted Linear Programming Model (CLPM)*:

$$\begin{aligned}
&\min \sum_{(i_{(1)},i_{(2)})\in S} V(i_{(1)},i_{(2)})\\
&\text{s.t. } V(i_{(1)},i_{(2)}) \geq r_1(i_{(1)},i_{(2)};a_{(1)}) + V(j_{(1)},i_{(2)}) \quad \forall (i_{(1)},i_{(2)}) \in S, a_{(1)} \in A_1(i_{(1)}), j_{(1)} = H_1(i_{(1)},a_{(1)})
\end{aligned} \tag{3.4}$$

$$\begin{aligned}
&V(i_{(1)},i_{(2)}) \geq r_2(i_{(1)},i_{(2)};a_{(2)}) + \beta \sum_{j_{(2)}\in S_2} P(i_{(1)},j_{(2)}|i_{(1)},i_{(2)};a_{(2)})V(i_{(1)},j_{(2)})\\
&\qquad \forall (i_{(1)},i_{(2)}) \in S, \quad a_{(2)} \in A_2(i_{(2)})
\end{aligned} \tag{3.5}$$

In the following section, we show that these two linear programming models obtain the same optimal solution.

4. Theoretical analysis of the TLPM model and the CLPM model

4.1. Property of the solutions to the TLPM model and the CLPM model

In this section, we investigate the property of the TLPM model and the CLPM model, and reveal the relation of the feasible solutions or the optimal solutions of the two linear programming models. The following theorem reveals the relation of the feasible solutions of the two linear programming models.

Theorem 4.1. *If V is a feasible solution to the CLPM model, then V is also a feasible solution to the TLPM model.*

Proof. Without loss of generality, let V denote $V(i_{(1)},i_{(2)})((i_{(1)},i_{(2)}) \in S)$. Since V is a feasible solution to the CLPM model, it satisfies constraints (3.4) and (3.5). From constraint (3.5), we have

$$V(j_{(1)},i_{(2)}) \geq r_2(j_{(1)},i_{(2)};a_{(2)}) + \beta \sum_{j_{(2)}\in S_2} P(j_{(1)},j_{(2)}|j_{(1)},i_{(2)};a_{(2)})V(j_{(1)},j_{(2)}) \quad \forall\, (j_{(1)},i_{(2)}) \in S, \quad a_{(2)} \in A_2(i_{(2)}) \tag{4.1}$$

From inequalities (3.4) and (4.1), we obtain inequality (3.3), *i.e.* V satisfies constraint (3.3). Hence, V is a feasible solution to the TLPM model.

The following theorem shows that under certain conditions, the optimal solution to the TLPM model is a feasible solution to the CLPM model. □

Theorem 4.2. *Assume that V^* is the optimal solution to the TLPM model. If the following two conditions are satisfied, then V^* is a feasible solution to the CLPM model.*

(1) For all sub-states $i_{(1)} \in S_1$ and $j_{(1)} \in S_1$, they are accessible to each other. That is, there exists two sub-actions $a_{(1)} \in A_1(i_{(1)})$ and $a_{(1)}^{(j)} \in A_1(j_{(1)})$ such that $j_{(1)} = H_1(i_{(1)}, a_{(1)})$ and $i_{(1)} = H_1(j_{(1)}, a_{(1)}^{(j)})$.

(2) For arbitrary state $(i_{(1)}, i_{(2)}) \in S$ and arbitrary action $a_{(1)}^{(1)} \in A_1(i_{(1)})$, $a_{(1)} \in A_1(i_{(1)})$ and $a_{(1)}^{(*)} \in A_1(j_{(1)})$, if $j_{(1)} = H_1(i_{(1)}, a_{(1)})$ and $H_1(i_{(1)}, a_{(1)}^{(1)}) = H_1(j_{(1)}, a_{(1)}^{(*)})$, then

$$r_1(i_{(1)}, i_{(2)}; a_{(1)}^{(1)}) \geq r_1(i_{(1)}, i_{(2)}; a_{(1)}) + r_1(j_{(1)}, i_{(2)}; a_{(1)}^{(*)}). \tag{4.2}$$

Proof. (1) Without loss of generality, let V^* denote $V^*(i_{(1)}, i_{(2)})((i_{(1)}, i_{(2)}) \in S)$. We first show that V^* satisfies constraint (3.4) in the CLPM model. Since V^* is the optimal solution to the TLPM model, it is the optimal value function of the corresponding CMDP process. According to Bellman optimality equation, we have

$$V^*(j_{(1)}, i_{(2)}) = \max_{a_{(1)} \in A_1(j_{(1)}), a_{(2)} \in A_2(i_{(2)})} \left\{ \begin{array}{l} r_1(j_{(1)}, i_{(2)}; a_{(1)}) + r_2(H_1(j_{(1)}, a_{(1)}), i_{(2)}; a_{(2)}) \\ +\beta \sum\limits_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a_{(1)}), j_{(2)} | j_{(1)}, i_{(2)}; a_{(1)}, a_{(2)}) V^*(H_1(j_{(1)}, a_{(1)}), j_{(2)}) \end{array} \right\}$$
$$\forall (j_{(1)}, i_{(2)}) \in S. \tag{4.3}$$

From equation (2.1), we have

$$P(H_1(j_{(1)}, a_{(1)}), j_{(2)} | j_{(1)}, i_{(2)}; a_{(1)}, a_{(2)}) = P(H_1(j_{(1)}, a_{(1)}), j_{(2)} | H_1(j_{(1)}, a_{(1)}), i_{(2)}; a_{(2)}). \tag{4.4}$$

It follows from equations (4.3) and (4.4) that

$$V^*(j_{(1)}, i_{(2)}) = \max_{a_{(1)} \in A_1(j_{(1)}), a_{(2)} \in A_2(i_{(2)})} \left\{ \begin{array}{l} r_1(j_{(1)}, i_{(2)}; a_{(1)}) + r_2(H_1(j_{(1)}, a_{(1)}), i_{(2)}; a_{(2)}) \\ +\beta \sum\limits_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a_{(1)}), j_{(2)} | H_1(j_{(1)}, a_{(1)}), i_{(2)}; a_{(2)}) \\ V^*(H_1(j_{(1)}, a_{(1)}), j_{(2)}) \end{array} \right\}$$
$$\forall (j_{(1)}, i_{(2)}) \in S. \tag{4.5}$$

For convenience, let U denote the number of sub-actions in $A_1(j_{(1)})$ and W denote the number of sub-actions in $A_2(i_{(2)})$. Let $a_{(1)}^u (1 \leq u \leq U)$ denote the uth sub-action in $A_1(j_{(1)})$ and $a_{(2)}^w (1 \leq w \leq W)$ denote the wth sub-action in $A_2(i_{(2)})$. Thus, it follows from equation (4.5) that

$$V^*(j_{(1)}, i_{(2)}) = \max_{1 \leq u \leq U, 1 \leq w \leq W} \left\{ \begin{array}{l} r_1(j_{(1)}, i_{(2)}; a_{(1)}^u) + r_2(H_1(j_{(1)}, a_{(1)}^u), i_{(2)}; a_{(2)}^w) \\ +\beta \sum\limits_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a_{(1)}^u), j_{(2)} | H_1(j_{(1)}, a_{(1)}^u), i_{(2)}; a_{(2)}^w) V^*(H_1(j_{(1)}, a_{(1)}^u), j_{(2)}) \end{array} \right\}$$

$$= \max \left\{ \begin{array}{l} \max\limits_{1 \leq w \leq W} \left\{ \begin{array}{l} r_1(j_{(1)}, i_{(2)}; a_{(1)}^1) + r_2(H_1(j_{(1)}, a_{(1)}^1), i_{(2)}; a_{(2)}^w) \\ +\beta \sum\limits_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a_{(1)}^1), j_{(2)} | H_1(j_{(1)}, a_{(1)}^1), i_{(2)}; a_{(2)}^w) V^*(H_1(j_{(1)}, a_{(1)}^1), j_{(2)}) \end{array} \right\}, \\ \max\limits_{1 \leq w \leq W} \left\{ \begin{array}{l} r_1(j_{(1)}, i_{(2)}; a_{(1)}^2) + r_2(H_1(j_{(1)}, a_{(1)}^2), i_{(2)}; a_{(2)}^w) \\ +\beta \sum\limits_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a_{(1)}^2), j_{(2)} | H_1(j_{(1)}, a_{(1)}^2), i_{(2)}; a_{(2)}^w) V^*(H_1(j_{(1)}, a_{(1)}^2), j_{(2)}) \end{array} \right\}, \\ \cdots \\ \max\limits_{1 \leq w \leq W} \left\{ \begin{array}{l} r_1(j_{(1)}, i_{(2)}; a_{(1)}^U) + r_2(H_1(j_{(1)}, a_{(1)}^U), i_{(2)}; a_{(2)}^w) \\ +\beta \sum\limits_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a_{(1)}^U), j_{(2)} | H_1(j_{(1)}, a_{(1)}^U), i_{(2)}; a_{(2)}^w) V^*(H_1(j_{(1)}, a_{(1)}^U), j_{(2)}) \end{array} \right\} \end{array} \right\}$$

$$\forall (j_{(1)}, i_{(2)}) \in S. \tag{4.6}$$

Since $r_1(j_{(1)}, i_{(2)}; a^u_{(1)})(1 \le u \le U)$ is independent of $a^w_{(2)}(1 \le w \le W)$, it follows that

$$\begin{aligned}
&V^*(j_{(1)}, i_{(2)}) \\
&= \max \left\{ \begin{array}{l}
r_1(j_{(1)}, i_{(2)}; a^1_{(1)}) + \max\limits_{1\le w \le W} \left\{ \begin{array}{l} r_2(H_1(j_{(1)}, a^1_{(1)}), i_{(2)}; a^w_{(2)}) \\ +\beta \sum\limits_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a^1_{(1)}), j_{(2)} | H_1(j_{(1)}, a^1_{(1)}), i_{(2)}; a^w_{(2)}) V^*(H_1(j_{(1)}, a^1_{(1)}), j_{(2)}) \end{array} \right\}, \\
r_1(j_{(1)}, i_{(2)}; a^2_{(1)}) + \max\limits_{1\le w \le W} \left\{ \begin{array}{l} r_2(H_1(j_{(1)}, a^2_{(1)}), i_{(2)}; a^w_{(2)}) \\ +\beta \sum\limits_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a^2_{(1)}), j_{(2)} | H_1(j_{(1)}, a^2_{(1)}), i_{(2)}; a^w_{(2)}) V^*(H_1(j_{(1)}, a^2_{(1)}), j_{(2)}) \end{array} \right\}, \\
\cdots \\
r_1(j_{(1)}, i_{(2)}; a^U_{(1)}) + \max\limits_{1\le w \le W} \left\{ \begin{array}{l} r_2(H_1(j_{(1)}, a^U_{(1)}), i_{(2)}; a^w_{(2)}) \\ +\beta \sum\limits_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a^U_{(1)}), j_{(2)} | H_1(j_{(1)}, a^U_{(1)}), i_{(2)}; a^w_{(2)}) V^*(H_1(j_{(1)}, a^U_{(1)}), j_{(2)}) \end{array} \right\}
\end{array} \right\} \\
&= \max_{1 \le u \le U} \left\{ r_1(j_{(1)}, i_{(2)}; a^u_{(1)}) + \max_{1\le w \le W} \left\{ \begin{array}{l} r_2(H_1(j_{(1)}, a^u_{(1)}), i_{(2)}; a^w_{(2)}) \\ +\beta \sum\limits_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a^u_{(1)}), j_{(2)} | H_1(j_{(1)}, a^u_{(1)}), i_{(2)}; a^w_{(2)}) \\ V^*(H_1(j_{(1)}, a^u_{(1)}), j_{(2)}) \end{array} \right\} \right\} \\
&= \max_{a_{(1)} \in A_1(j_{(1)})} \left\{ r_1(j_{(1)}, i_{(2)}; a_{(1)}) + \max_{a_{(2)} \in A_2(i_{(2)})} \left\{ \begin{array}{l} r_2(H_1(j_{(1)}, a_{(1)}), i_{(2)}; a_{(2)}) \\ +\beta \sum\limits_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a_{(1)}), j_{(2)} | H_1(j_{(1)}, a_{(1)}), i_{(2)}; a_{(2)}) \\ V^*(H_1(j_{(1)}, a_{(1)}), j_{(2)}) \end{array} \right\} \right\} \\
&\quad \forall (j_{(1)}, i_{(2)}) \in S.
\end{aligned} \tag{4.7}$$

Suppose

$$v(j_{(1)}, i_{(2)}; a_{(1)}) = \max_{a_{(2)} \in A_2(i_{(2)})} \left\{ \begin{array}{l} r_2(H_1(j_{(1)}, a_{(1)}), i_{(2)}; a_{(2)}) \\ +\beta \sum\limits_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a_{(1)}), j_{(2)} | H_1(j_{(1)}, a_{(1)}), i_{(2)}; a_{(2)}) V^*(H_1(j_{(1)}, a_{(1)}), j_{(2)}) \end{array} \right\}, \tag{4.8}$$

and

$$a^{(*)}_{(1)} = \operatorname*{arg\,max}_{a_{(1)} \in A_1(j_{(1)})} \left\{ r_1(j_{(1)}, i_{(2)}; a_{(1)}) + v(j_{(1)}, i_{(2)}; a_{(1)}) \right\}, \tag{4.9}$$

thus it follows from equation (4.7) that

$$\begin{aligned}
V^*(j_{(1)}, i_{(2)}) = r_1(j_{(1)}, i_{(2)}; a^{(*)}_{(1)}) + &\max_{a_{(2)} \in A_2(i_{(2)})} \{ r_2(H_1(j_{(1)}, a^{(*)}_{(1)}), i_{(2)}; a_{(2)}) \\
&+ \beta \sum_{j_{(2)} \in S_2} P(H_1(j_{(1)}, a^{(*)}_{(1)}), j_{(2)} | H_1(j_{(1)}, a^{(*)}_{(1)}), i_{(2)}; a_{(2)}) V^*(H_1(j_{(1)}, a^{(*)}_{(1)}), j_{(2)}) \} \\
&\forall (j_{(1)}, i_{(2)}) \in S
\end{aligned} \tag{4.10}$$

Suppose $k_{(1)} = H_1(j_{(1)}, a^{(*)}_{(1)})$. According to condition (2.1), for all sub-states $i_{(1)} \in S_1$ and $k_{(1)} \in S_1$, there exists an action $a^{(1)}_{(1)} \in A_1(i_{(1)})$ such that $k_{(1)} = H_1(i_{(1)}, a^{(1)}_{(1)})$.

V^* is a feasible solution to the TLPM model, it satisfies constraint (3.3), thus we have

$$\begin{aligned}
V^*(i_{(1)}, i_{(2)}) \ge r_1(i_{(1)}, i_{(2)}; a^{(1)}_{(1)}) + r_2(k_{(1)}, i_{(2)}; a_{(2)}) + \beta \sum_{j_{(2)} \in S_2} P(k_{(1)}, j_{(2)} | k_{(1)}, i_{(2)}; a_{(2)}) V^*(k_{(1)}, j_{(2)}) \\
\forall (i_{(1)}, i_{(2)}) \in S, \quad a^{(1)}_{(1)} \in A_1(i_{(1)}), \quad a_{(2)} \in A_2(i_{(2)})
\end{aligned} \tag{4.11}$$

where $k_{(1)} = H_1(i_{(1)}, a^{(1)}_{(1)})$.

Suppose $H_1(i_{(1)}, a_{(1)}) = j_{(1)}$. Since $H_1(i_{(1)}, a^{(1)}_{(1)}) = k_{(1)}$ and $H_1(j_{(1)}, a^{(*)}_{(1)}) = k_{(1)}$, it follows from condition (2) that

$$r_1(i_{(1)}, i_{(2)}; a^{(1)}_{(1)}) \ge r_1(i_{(1)}, i_{(2)}; a_{(1)}) + r_1(j_{(1)}, i_{(2)}; a^{(*)}_{(1)}) \quad \forall (i_{(1)}, i_{(2)}) \in S. \tag{4.12}$$

It follows from equations (4.11) and (4.12) that

$$V^*(i_{(1)},i_{(2)}) \geq r_1(i_{(1)},i_{(2)};a_{(1)}) + r_1(j_{(1)},i_{(2)};a_{(1)}^{(*)}) + r_2(k_{(1)},i_{(2)};a_{(2)}) + \beta \sum_{j_{(2)}\in S_2} P(k_{(1)},j_{(2)}|k_{(1)},i_{(2)};a_{(2)})V^*(k_{(1)},j_{(2)})$$
$$\forall (i_{(1)},i_{(2)}) \in S, a_{(1)} \in A_1(i_{(1)}), \quad a_{(2)} \in A_2(i_{(2)}), \quad j_{(1)} = H_1(i_{(1)},a_{(1)}) \tag{4.13}$$

Hence,

$$\begin{aligned} V^*(i_{(1)},i_{(2)}) \geq r_1(i_{(1)},i_{(2)};a_{(1)}) + r_1(j_{(1)},i_{(2)};a_{(1)}^{(*)}) + \max_{a_{(2)}\in A_2(i_{(2)})} \{ r_2(k_{(1)},i_{(2)};a_{(2)}) \\ + \beta \sum_{j_{(2)}\in S_2} P(k_{(1)},j_{(2)}|k_{(1)},i_{(2)};a_{(2)})V^*(k_{(1)},j_{(2)})\} \\ \forall (i_{(1)},i_{(2)}) \in S, a_{(1)} \in A_1(i_{(1)}), \quad j_{(1)} = H_1(i_{(1)},a_{(1)}) \end{aligned} \tag{4.14}$$

It follows from equations (4.10) and (4.14) that

$$V^*(i_{(1)},i_{(2)}) \geq r_1(i_{(1)},i_{(2)};a_{(1)}) + V^*(j_{(1)},i_{(2)}) \quad \forall (i_{(1)},i_{(2)}) \in S, a_{(1)} \in A_1(i_{(1)}), \quad j_{(1)} = H_1(i_{(1)},a_{(1)}) \tag{4.15}$$

Thus, V^* satisfies constraint (3.4) in the CLPM model.

(2) Next, we show that V^* satisfies constraint (3.5) in the CLPM model. Without loss of generality, suppose $a_{(1)}^{(0)}$ is a dummy action such that $H_1(i_{(1)},a_{(1)}^{(0)}) = i_{(1)}$ and $r_1(i_{(1)},i_{(2)};a_{(1)}^{(0)}) = 0$. Since V^* is the optimal solution to the TLPM model, V^* satisfies constraint (3.3). Thus,

$$V^*(i_{(1)},i_{(2)}) \geq r_1(i_{(1)},i_{(2)};a_{(1)}^{(0)}) + r_2(i_{(1)},i_{(2)};a_{(2)}) + \beta \sum_{j_{(2)}\in S_2} P(i_{(1)},j_{(2)}|i_{(1)},i_{(2)};a_{(2)})V^*(i_{(1)},j_{(2)}) \tag{4.16}$$
$$\forall (i_{(1)},i_{(2)}) \in S, \quad a_{(2)} \in A_2(i_{(2)}) \tag{4.17}$$

Since $H_1(i_{(1)},a_{(1)}^{(0)}) = i_{(1)}$ and $r_1(i_{(1)},i_{(2)};a_{(1)}^{(0)}) = 0$, it follows from inequality (4.17) that

$$\begin{aligned} V^*(i_{(1)},i_{(2)}) &\geq 0 + r_2(i_{(1)},i_{(2)};a_{(2)}) + \beta \sum_{j_{(2)}\in S_2} P(i_{(1)},j_{(2)}|i_{(1)},i_{(2)};a_{(2)})V^*(i_{(1)},j_{(2)}) \\ &= r_2(i_{(1)},i_{(2)};a_{(2)}) + \beta \sum_{j_{(2)}\in S_2} P(i_{(1)},j_{(2)}|i_{(1)},i_{(2)};a_{(2)})V^*(i_{(1)},j_{(2)}) \\ &\forall (i_{(1)},i_{(2)}) \in S, \quad a_{(2)} \in A_2(i_{(2)}) \end{aligned} \tag{4.18}$$

Thus, V^* satisfies constraint (3.5) in the CLPM model. From part (1), V^* also satisfies constraint (3.4) in the CLPM model. Hence, V^* satisfies all the constraints in the CLPM model. Consequently, V^* is a feasible solution to the CLPM model. □

According to Theorems 4.1 and 4.2, we can show that the TLPM model and the CLPM model have the same optimal solution.

Theorem 4.3. *Assume that V is the optimal solution to the CLPM model. If the two conditions in Theorem 4.2 are satisfied, then V is also the optimal solution to the TLPM model.*

Proof. By contradiction. Suppose V is not an optimal solution to the TLPM model. Since V is the optimal solution to the CLPM model, it is a feasible solution to the CLPM model. By Theorem 4.1, V is also a feasible solution to the TLPM model. Suppose V^* is the optimal solution to the TLPM model, then both V and V^* are feasible solutions to the TLPM model. According to the objective function in the TLPM model, we obtain that

$$\sum_{(i_{(1)},i_{(2)})\in S} V(i_{(1)},i_{(2)}) > \sum_{(i_{(1)},i_{(2)})\in S} V^*(i_{(1)},i_{(2)}). \tag{4.19}$$

TABLE 1. Experiment results with respect to the discount factor β.

β		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Objective function value		158	179	207	274	396	577	717	1084	1664
Running time (s)	TLPM Model	0.269	0.287	0.303	0.252	0.261	0.297	0.268	0.275	0.261
	CLPM Model	0.158	0.164	0.170	0.155	0.156	0.173	0.163	0.152	0.150

Since the two conditions in Theorem 4.2 are satisfied, V^* is a feasible solution to the CLPM model according to Theorem 4.2. If follows from inequality (4.19) that for the CLPM model, V^* is a better solution than V. That is, V is not the optimal solution to the CLPM model, which contradicts with the assumption in Theorem 4.3. Consequently, V is the optimal solution to the TLPM model.

According to Theorem 4.3, the TLPM model and the CLPM model obtain the same optimal solution. Because the optimal solution to the TLPM model is the optimal state value function of the corresponding CMDP process, the optimal solution to the CLPM model is also the optimal state value function of the CMDP process, thus we can solve the CLPM model to obtain the optimal state values and then obtain the optimal control policy by equation (3.1). □

4.2. Comparison of the TLPM model and the CLPM model

In the following, we analyze the scale of the TLPM model and the CLPM model. The number of decision variables in the TLPM model and the CLPM model are both $|S|$, where S is the state space and $|S|$ denotes the number of states in the state space. The number of constraints in the TLPM model (constraint (3.3)) is $|S||A_1(i_{(1)})||A_2(i_{(2)})|$, where $|A_1(i_{(1)})|$ denotes the number of sub-actions in sub-action space $A_1(i_{(1)})$ and $|A_2(i_{(2)})|$ denotes the number of sub-actions in sub-action space $A_2(i_{(2)})$. However, the number of constraints in constraint (3.4) is $|S||A_1(i_{(1)})|$ and the number of constraints in constraint (3.5) is $|S||A_2(i_{(2)})|$, thus the total number of constraints in the CLPM model is $|S|(|A_1(i_{(1)})| + |A_2(i_{(2)})|)$. Therefore, theoretically the CLPM model needs fewer constraints than the TLPM model. The number of constraints in the TLPM model is $|A_1(i_{(1)})||A_2(i_{(2)})|/(|A_1(i_{(1)})|+|A_2(i_{(2)})|)$ times of that in the CLPM model. Hence, solving the CLPM model needs less computational effort and smaller memory size than solving the TLPM model. The comparison computational experiments of the two models are conducted in the next section.

5. COMPUTATIONAL EXPERIMENTS

In this section, we conduct computational experiments with IBM ILOG OPL-CPLEX software to examine the performance of the TLPM model and the CLPM model for solving a CMDP process. In the experiment instances, the probability matrixes and the reward parameters are known. For each $(a_{(1)}, a_{(2)})$ combination, we set a specific probability matrix of state transition. Let $n_1 = |S_1|$, $n_2 = |S_2|$, $k_1 = |A_1(i_{(1)})|$, $k_2 = |A_2(i_{(2)})|$. That is, n_1, n_2, k_1 and k_2 denote the size of $|S_1|$, $|S_2|$, $|A_1(i_{(1)})|$ and $|A_2(i_{(2)})|$, respectively. First we examine the impact of the discount factor β on the running times the two models need. We set $n_1 = n_2 = k_1 = k_2 = 5$ and observe the objective function values and the running times of the two models when β takes 0.1, 0.2,..., 0.9, respectively. As shown in Table 1, the two models obtain the same optimal objective function value for each β value. The optimal objective function value increases quickly with increasing of β. However, variation of β value has slightly impact on the running times of the two models. The CLPM model needs much shorter running time than the TLPM model for any given β value.

To investigate and compare the performance of the two models, we conduct more computational experiments with problems of different scales. Let β take the fixed value 0.9 and the four parameters n_1, n_2, k_1 and k_2 take different values varying from 5 to 25. Then we observe the objective function values, the running times and memory usage of the two models. For each value combination of (n_1, n_2, k_1, k_2) parameters, we conduct 50 instances with different datasets and take the average of the experiment results. The experiment results

TABLE 2. Experiment results with different value combinations of (n_1, n_2, k_1, k_2).

n_1	n_2	k_1	k_2	Number of Variables	Number of Constraints		Objective Function Value	Running Time (s)		Memory Usage (M)	
					TLPM Model	CLPM Model		TLPM Model	CLPM Model	TLPM Model	CLPM Model
5	5	5	5	25	625	250	1664.05	0.261	0.150	1.211	0.644
6	6	6	6	36	1296	432	2586.12	0.354	0.183	0.561	0.268
7	7	7	7	49	2401	686	3801.51	0.422	0.201	6.311	1.891
8	8	8	8	64	4096	1024	5418.36	0.525	0.228	6.453	1.531
9	9	9	9	81	6561	1458	7291.68	0.555	0.233	15.663	3.098
10	10	10	10	100	10000	2000	9435.39	0.650	0.265	15.191	2.590
11	11	11	11	121	14641	2662	12252.78	0.897	0.343	22.647	3.711
12	12	12	12	144	20736	3456	15381.29	0.935	0.330	30.684	4.906
13	13	13	13	169	28561	4394	18782.60	1.423	0.442	41.922	6.660
14	14	14	14	196	38416	5488	22881.79	1.819	0.543	57.848	9.078
15	15	15	15	225	50625	6750	27264.74	2.472	0.722	74.787	11.641
16	16	16	16	256	65536	8192	32550.78	3.407	0.961	92.867	14.285
17	17	17	17	289	83521	9826	38472.50	4.499	1.223	123.479	18.840
18	18	18	18	324	104976	11664	44546.30	6.101	1.636	147.613	22.503
19	19	19	19	361	130321	13718	51655.17	8.443	2.177	175.782	26.528
20	20	20	20	400	160000	16000	59204.97	11.691	2.964	227.988	32.490
21	21	21	21	441	194481	18522	67655.39	15.585	3.865	276.501	38.735
22	22	22	22	484	234256	21296	76574.07	20.691	4.985	314.052	42.881
23	23	23	23	529	279841	24334	86266.90	27.633	6.561	386.670	50.232
24	24	24	24	576	331776	27648	96807.66	36.866	8.640	472.273	59.145
25	25	25	25	625	390625	31250	108322.02	48.584	11.351	601.309	71.531

are shown as Table 2 As shown in Table 2, for arbitrary value combination of (n_1, n_2, k_1, k_2) parameters, the objective function values of the two models are identical, however, the running time of the TLPM model is much longer than that of the CLPM model and memory usage of the TLPM model is much larger than that of the CLPM model.

Let $q = n_1 = n_2 = k_1 = k_2$. To further investigate the computation efficiency and adaptability of the two models for different problem scales, we draw curves (Figs. 2 and 3) to reveal the variation of running times and memory usage of the two models with increasing of q value. As shown in Figure 2, though the running time monotonically increases with increasing of q value for both the two models, the TLPM model needs longer running time than the CLPM model for arbitrary q value and the running time of the TLPM model increases much faster than that of the CLPM model with increasing of q value. The increment speed of the running time of the TLPM model becomes faster and faster. The running time of the TLPM model grows from 0.261 s to 2.472 s while q value increases from 5 to 15. When q is larger than 15, the running time of the TLPM model grows rapidly. When q is equal to 25, the running time of the TLPM model reaches a considerable value of 48.584 s. However, the running times of the CLPM model are only 0.150 s, 0.722 s and 11.351 s, respectively when q takes 5, 15 and 25, respectively.

The shape of memory usage curve shown in Figure 3 is similar to the shape of running time curve shown in Figure 2. As shown in Figure 3, though memory usage monotonically increases with increasing of q value for both the two models, the TLPM model needs larger memory usage than the CLPM model for arbitrary q value and the memory usage of the TLPM model increases much faster than that of the CLPM model with increasing of q value. The increment speed of memory usage of the TLPM model becomes faster and faster. The memory usage of the TLPM model grows from 1.211 M to 74.787 M when q value increases from 5 to 15. When

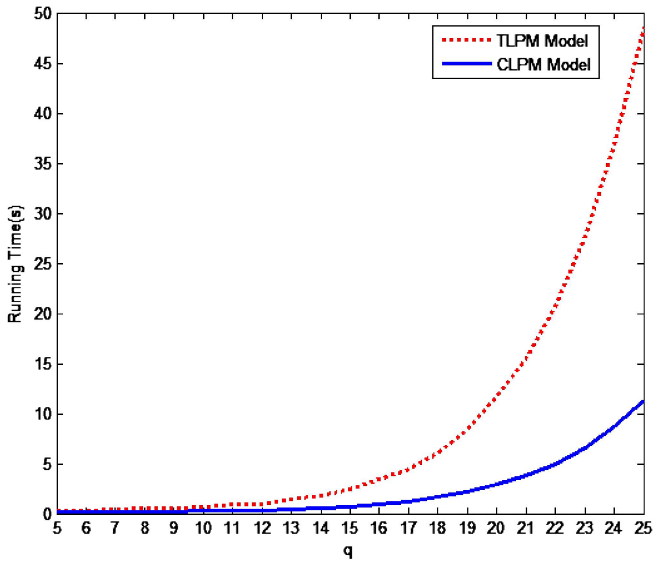

Figure 2. Variation of the running times of the two models with respect to q value.

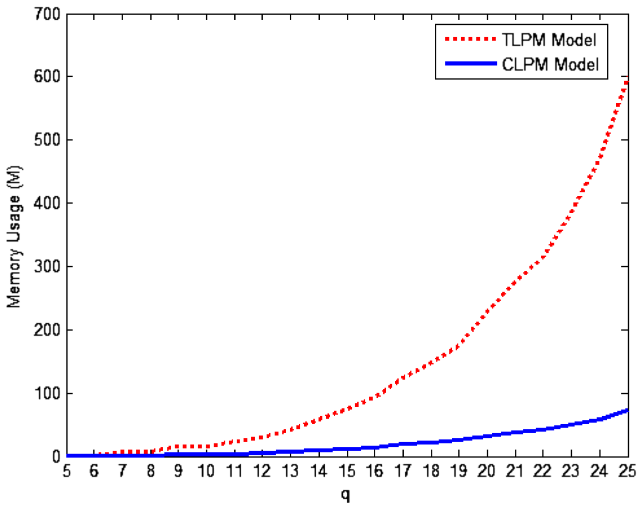

Figure 3. Variation of memory usage of the two models with respect to q value.

q is larger than 15, the memory usage of the TLPM model grows rapidly. When q is equal to 25, the memory usage of the TLPM model reaches a considerable value of 601.309 M. However, the memory usage of the CLPM model is only 0.644 M, 11.641 M and 71.531 M, respectively when q takes 5, 15 and 25, respectively.

Experimental results reveal that because the CLPM model generates much fewer constraints than the TLPM model for any specific problem, it needs much more computational effort to solve the TLPM model than the CLPM model in both computation time and memory usage. The growth of both running time and memory usage with increasing of problem size for the TLPM model is much faster than that of the CLPM model. Thus, the CLPM model is remarkably superior to the TLPM model in both time complexity and space complexity.

6. Conclusions

We study a time homogeneous discrete Markov decision process with composite actions (CMDP) which needs to make multiple decisions at each state. In this particular Markov decision process, the state variables are divided into two separable sets and a two-dimensional composite action is chosen at each decision epoch. To solve a Markov decision process with composite actions, we propose a novel linear programming model (CLPM model). We show that the CLPM model obtains the optimal state values of a CMDP process as the Traditional Linear Programming Model (TLPM model). We compare the number of variables and constraints of the CLPM model and the TLPM model. The number of constraints in the TLPM model is $|A_1(i_{(1)})||A_2(i_{(2)})|/(|A_1(i_{(1)})| + |A_2(i_{(2)})|)$ times of that in the CLPM model. We also conduct computational experiments to compare the running times and memory usage of the two models. The CLPM model outperforms the TLPM model in both time complexity and space complexity by theoretical analysis and computational experiments.

In this paper, a composite action taken in a CMDP process is composed of two dimensional sub-actions. For a CMDP process whose composite action is composed of three or more dimensional sub-actions, if it has the special extended structure as the Markov decision process illustrated in Section 2, its corresponding TLPM model and CLPM model can also be constructed similar to the models in Section 3.

Acknowledgements. This work is supported by Natural Science Foundation of Guangdong Province, China (No. 2015A030313649), Science and Technology Planning Project of Guangdong Province, China (No. 2015A010103021), National Natural Science Foundation of China (Grant No. 61703102).

References

[1] H.S. Ahn, I. Duenyas and M.E. Lewis, The optimal control of a two-stage tandem queueing system with flexible servers. *Prob. Eng. Inf Sci.* **16** (2002) 453–469.

[2] H.S. Ahn and R. Righter, Multi-actor Markov decision processes. *J. Appl. Prob.* **42** (2005) 15–26.

[3] S. Andradóttir and H. Ayhan, Throughput maximization for tandem lines with two stations and flexible servers. *Oper. Res.* **53** (2005) 516–531.

[4] J.J. Hasenbein and B. Kim, Throughput maximization for two station tandem systems: a proof of the Andradóttir–Ayhan conjecture. *Queue. Syst.* **67** (2011) 365–386.

[5] D.L. Kaufman, H.S. Ahn and M.E. Lewis, On the introduction of an agile, temporary workforce into a tandem queueing system. *Queue. Syst.* **51** (2005) 135–171.

[6] C. Kim and S. Dudin, Priority tandem queueing model with admission control. *Comput. Ind. Eng.* **61** (2011) 131–140.

[7] E. Kırkızlar, S. Andradóttir and H. Ayhan, Robustness of efficient server assignment policies to service time distributions in finite-buffered lines. *Nav. Res. Logist.* **57** (2010) 563–582.

[8] E. Kırkızlar, S. Andradóttir and H. Ayhan, Flexible servers in understaffed tandem lines. *Prod. Oper. Manage.* **21** (2012) 761–777.

[9] D.G. Pandelis, Optimal control of flexible servers in two tandem queues with operating costs. *Prob. Eng. Inf. Sci.* **22** (2008) 107–131.

[10] D.G. Pandelis, Optimal stochastic scheduling of two interconnected queues with varying service rates. *Oper. Res. Lett.* **36** (2008) 492–495.

[11] D.G. Pandelis, Markov decision processes with multidimensional action spaces. *Eur. J. Oper. Res.* **200** (2010) 625–628.

[12] M.L. Puterman, Markov Decision Processes: Discrete Dynamic Stochastic Programming. John Wiley & Sons, Inc., New York, NY (1994).

[13] C.H. Wu, D.G. Down and M.E. Lewis, Heuristics for allocation of reconfigurable resources in a serial line with reliability considerations. *IIE Trans.* **40** (2008) 595–611.

[14] J. Yang, Nested Markov decision framework for coordinating pavement improvement with capacity expansion. *J. Transp. Eng.* **138** (2012) 387–394.