Computationally Efficient Approximate Dynamic Programming for Multi-site Production Capacity Planning with Uncertain Demands

Chen-Yang Cheng

Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 10608, Taiwan; Email: cycheng@ntut.edu.tw

Pourya Pourhejazy

Department of Industrial Engineering, UiT- The Arctic University of Norway, Lodve Langsgate 2, Narvik 8514, Norway; Email: pourya.pourhejazy@uit.no

Tzu-Li Chen

Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 10608, Taiwan; Email: chentzuli@gmail.com

ABSTRACT

With globalization and rapid technological-economic development accelerating the market dynamics, consumers' demand is becoming more volatile and diverse. In this situation, capacity adjustment as an operational strategic decision plays a major role to ensure supply chain responsiveness while maintaining costs at a reasonable norm. This study contributes to the literature by developing computationally efficient approximate dynamic programming approaches for production capacity planning considering uncertainties and demand interdependence in a multi-factory multi-product supply chain setting. For this purpose, the *k*-Nearest-Neighbor-based Approximate Dynamic Programming (*k*-ADP) and the Rolling-Horizon-based Approximate Dynamic Programming (*R*-ADP) are developed to enable real-time decision support while ensuring the robustness of the outcomes in stochastic decision environments. Given the market volatilities in the Thin Film Transistor-Liquid Crystal Display (TFT-LCD) industry, a real case from this sector is investigated to evaluate the applicability of the developed approach and provide insights for other industry situations. The developed method is less complex to implement, and numerical experiments showed that it is also computationally more efficient compared to Stochastic Dynamic Programming (SDP).

Keywords: Production management, Capacity Planning, Approximate Dynamic Programming, Stochastic Dynamic Programming, Optimization.

1. Introduction

Globalization has exacerbated market volatilities and added to supply chain management complexities. Modern industries are experiencing exponential growth in demand and operational risks, particularly driven by rapid technological and economic development. Well-informed supply and demand management decisions are of paramount significance to ensure a timely response to market volatilities. Increasing production capacity, inventory level, supply source redundancy, and operational flexibility are the major supply chain management strategies to mitigate operational risks (Chopra and Meindl 2015), which are not one-size-fits-all decisions and require various considerations. As a prime example, building redundant operational capacity improves the supply chains' resilience (Pourhejazy et al., 2017), but may not be feasible in capital-intensive industries like high-tech. Decision support systems are required to ensure well-informed capacity planning decisions that account for short- and long-term considerations.

Supply chain capacity planning is not limited to the strategic level and can be made in forms of tactical and operational course of managerial actions (Martinez-Costa et al. 2014), like adding auxiliary production tools (Chen et al. 2013), among other capacity planning strategies (Ceryan and Koren 2009). Changes in the current state of the controllable and uncontrollable variables influence their future value and the performance of the system as a whole; this kind of sequential effect (Pourhejazy et al., 2020) cannot be effectively captured in the static models. Dynamic capacity expansion models best determine policies for optimal timing and the extent of capacity adjustment with the overall goal of maximizing the expected profit (Wu et al. 2005). The decisions of when and how much capacity to build are vital to staying competitive in highly uncertain environments (Wu and Chuang 2010).

Dynamic programming has been applied in various contexts, like in project management (Choi et al. 2004), remanufacturing (Li et al. 2009), among other examples (La Torre et al. 2019; Tarek et al. 2020); its applications for production capacity planning are relatively limited (Choi et al. 2006; Katanyukul et al. 2011). From the existing studies, backlog (Kingsman 1997) and workload control (Kingsman 2000) were investigated as strategic dynamic capacity planning decisions. (Bunn and Oliveira 2016) analyzed active trading of production facilities, analyzing dynamic capacity planning from an economic viewpoint and provided the theoretical basis for assets valuation and trading based on the strategic slack concept. System dynamic approaches were also employed to study the main sources of complexity (Deif and ElMaraghy 2009) and analyze capacity planning policies for remanufacturing facilities in reverse supply chains (Vlachos et al. 2007; Georgiadis 2013). (Pehlivan et al. 2014) proposed to integrate a service level constraint to dynamic capacity planning and location decisions. Dynamic capacity planning has also seen some development for cloud services (Alasaly et al. 2015).

Dynamic capacity planning using simulation models was the first time investigated by (Atherton 1989). Since then, different simulation-based capacity planning methods have been investigated to improve solutions consistency (Seidel et al. 2019). (Pratikakis et al. 2006) developed a real-time adaptive dynamic programming method for planning and scheduling. (Pratikakis et al. 2010) studied Approximate Dynamic Programming (ADP) method to address the curse-of-dimensionality associated with multistage capacity decision problems. More

recently, (Lin et al. 2014) developed a Stochastic Dynamic Programming (SDP) approach for multi-site capacity planning in thin-film transistor liquid crystal display (TFT-LCD) manufacturing considering demand uncertainty. Several patents have also been registered for dynamic capacity planning (Sureka 2019; Dubey 2020). (Hu et al. 2020) developed a two-stage dynamic capacity planning approach considering demand uncertainty for a single-site capacity planning for maintenance of agriculture machinery. In their method, a scheduling model is integrated with capacity allocation decisions and queuing theory is used to determine the operational parameters. In the most recent study, (Voelkel et al. 2020) developed an aggregation-based ADP for the periodic review of manufacturing systems considering random yield. They did not account for the stochasticity of capacity and demand transitions and the interdependence between demand in the early and late stages of the planning horizon. There also is a need for computationally efficient ADP methods that provide real-time decision support to facilitate the industry-scale applications of dynamic programming while addressing the above limitations for a more practical setting.

Inspired by the mentioned motive, the present manuscript introduces two simulation-based ADPs, namely, the *k*-Nearest-Neighbor-based Approximate Dynamic Programming (*k*-ADP) and the Rolling-Horizon-based Approximate Dynamic Programming (*R*-ADP) to contribute to the production capacity planning literature. The developed methods are benchmarked against the SDP method considering various problem specifications. Given high volatilities in TFT-LCD industries (Lin et al. 2011), a real case from this industry is used to inform similar or more stable industry situations. Additionally, large-scale examples are considered to examine the computational efficiency of the developed methods for industry-scale applications.

The rest of this manuscript is structured as follows. The proposed methods, *k*-ADP and *R*-ADP are detailed in Section 2. Numerical experiment and results analysis are provided in Section 3 followed by statistical analysis to verify the findings. Finally, the major findings, limitations of the study, and suggestions for future research are discussed in Section 4 to conclude this research work.

2. Proposed methods

Solving multi-period optimization problems using stochastic dynamic programming becomes prohibitive when real-world and complex situations consisting of many decision and state variables are addressed. In this situation, finding the best trajectory through full enumeration of the solutions is infeasible, particularly if reasonable computational time is a must for instantaneous decision analysis. This section presents two computationally efficient ADPs, named *k*-Nearest-Neighbor-based Approximate Dynamic Programming (*k*-ADP), and Rolling-Horizon-based Approximate Dynamic Programming (*R*-ADP), to address the production capacity planning decisions in stochastic environments. The Markov Chain Monte-Carlo simulation (MCMCS) is used to sample the demand trajectories and an approximation algorithm investigates the capacity expansion trajectories through partial-enumeration of the state space using Forward Induction. The research framework is illustrated in Figure 1 with its major elements described in the following sub-sections.

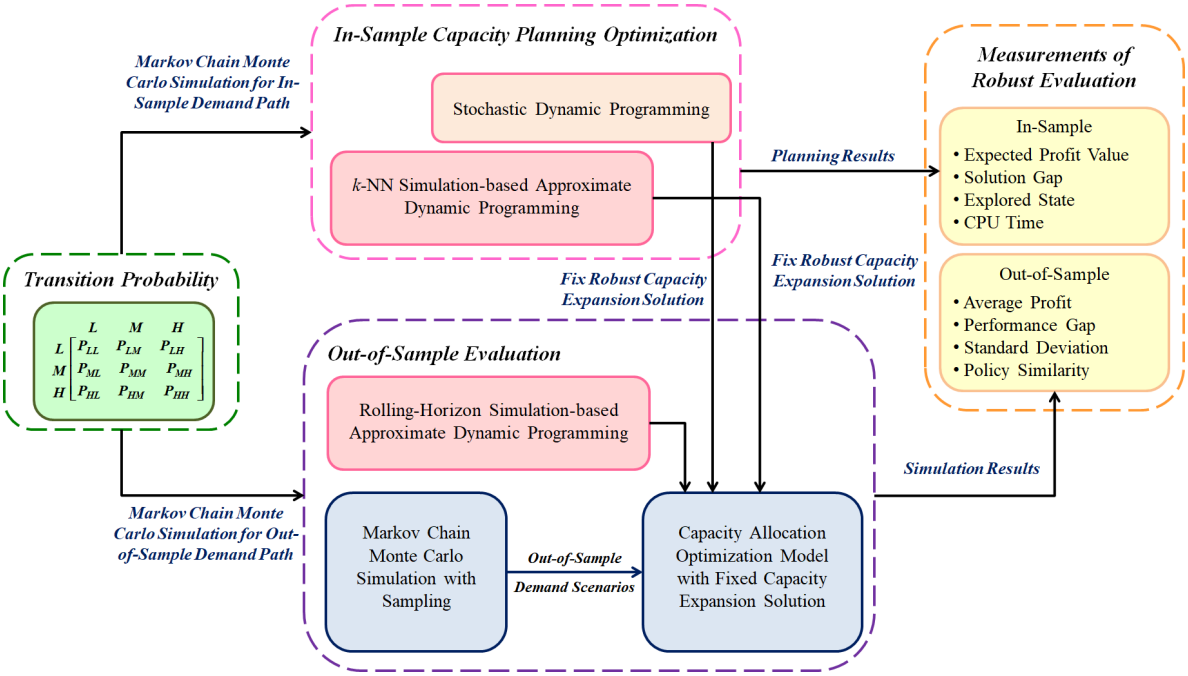


Figure 1. Research framework

2.1. Problem definition

To provide a formal definition of the problem, let consider the TFT-LCD supply chain. As shown in Figure 2, I array production sites produce glass substrate manufacturers to produce P product families, e.g., monitors and TVs. Given the forecasts based on historical demand data, a planning horizon of T months is considered for strategic capacity planning based on which the aggregate and master production plans drive the operations. With high implied demand uncertainties and interdependence between demand in the early and late stages of the planning horizon, Markovian demand, the supply-demand mismatch is a common phenomenon. This imbalance often results in lost sales or excessive investment in resources. Given the characteristics of the product, inventory holding costs may be high, hence, is not an option to manage supply-demand imbalance. In this situation, capacity reallocation and expansion decisions are major managerial tools to alleviate the above issues.

For TFT-LCD manufacturing, production quantities of each product group are constrained by two resources: bottleneck machines and auxiliary tools. While the bottleneck machine is a shared resource for all products, auxiliary tools are the product-group-specific resource. In the array process, the shared resource is the lithographic machine. The product-group-specific auxiliary tool, photomask, must be used by the lithographic machines for transferring a pattern on a substrate. Therefore, the production capacity of each array plant is determined by the number of bottleneck machines. This kind of capacity can be reallocated to the products of one factory. But the available capacity for producing each product model at each plant is determined by the number of auxiliary tools. This capacity is product-group-specific capacity and cannot be used by other products. The manufacturer can expand the production capacity of each array plant and the available capacity for producing each product model at each plant through purchasing the bottleneck machines and the auxiliary tool, respectively. Due to space limitations, it is difficult to purchase the bottleneck machines to expand the production capacity of the plant.

Consequently, this study only focuses on the investment of the auxiliary tool to expand the available capacity for producing each product model at each plant.

Given the actual demand of each product model at the beginning of the planning horizon, various demand scenarios, the production capacity of each array plant, and the available capacity for producing each product model at each plant, the problem at hand consists of production capacity allocation and expansion decisions to determine the best product mix and quantity at each plant in each period of the planning horizon such that the total profit is maximized. The expected value and variance of the forecasted demand, as well as demand interdependence in the early and late stages, are considered to account for the demand uncertainties in the model.

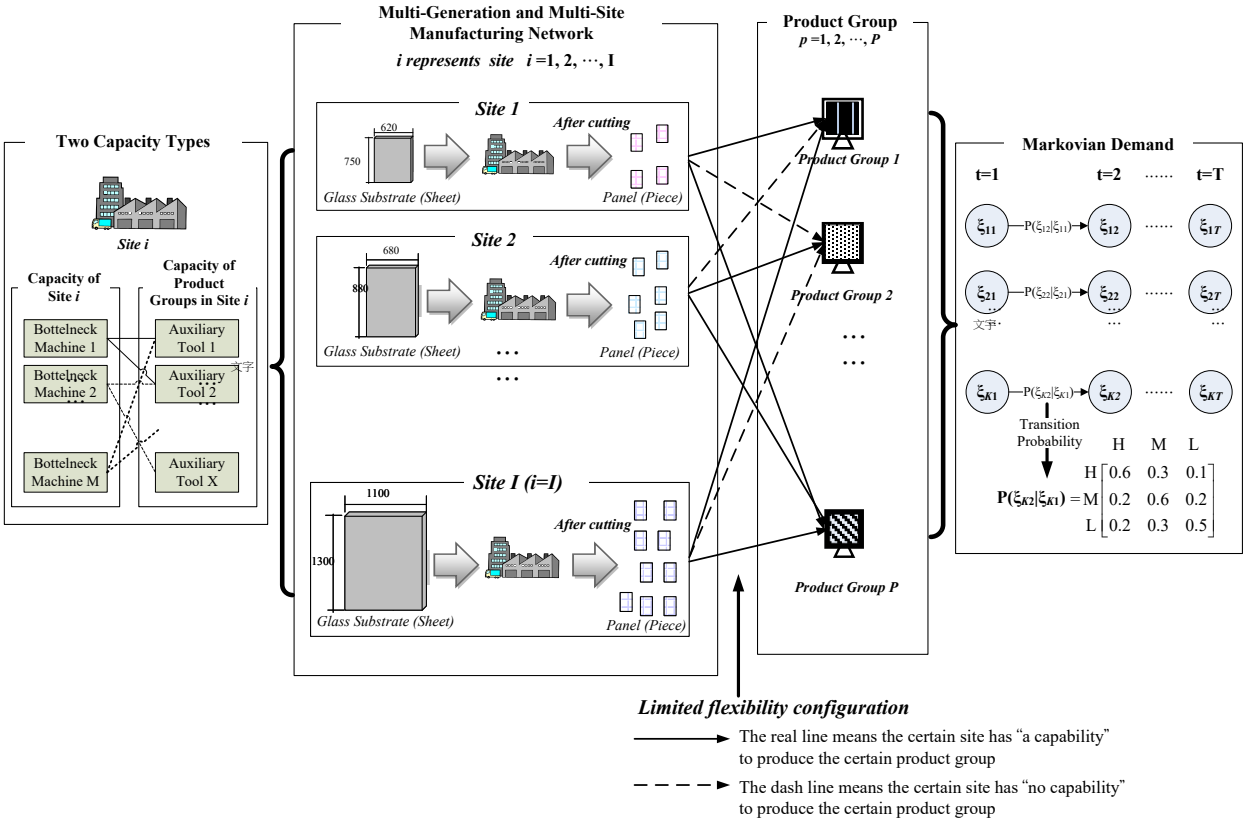


Figure 2. Schematic illustration of the studied multi-site production chain

The following information is known before starting the planning process: (1) distribution function of the forecasted demand and future scenarios of each product model within the planning horizon; (2) transition probability of demand scenarios; (3) the phase-out time, i.e., the time a product model is discontinued. (4) the primary production capacity of each array manufacturer; (5) the available capacity of each product family at each array manufacturer; (6) production yield rate and the economic cutting ratio of each product family at each manufacturing plant; (7) the allowed expansion capacity of each product family in each manufacturing plant, and the unit cost of purchasing the auxiliary tools.

Given the above inputs, dynamic capacity planning is about deciding which product, where, and when requires capacity expansion considering the current state of interest, i.e., the cumulative capacity expansion in each period and the current demand situation. On this basis,

the purchase of auxiliary tools in period t increases the production capacity of product family p. Capacity expansion decisions are strongly influenced by the investment cost, the demand uncertainties during the planning period, and the depreciation rate (Wu et al. 2005). Finite Markovian demand is considered to account for the uncertainties. Because the cost of capacity expansion is very high, in practice, the linear depreciation method is used to calculate the amortized capacity cost by considering the usable life of the auxiliary tool in the cost accounting. To account for the depreciation rate, one should keep in mind the products' life cycle, in particular the possibility of discontinuing them from the company's portfolio. Equation (1) represents the linear depreciation rate of the product. As illustrated in Figure 3, if product p won't be discontinued within the planning horizon, i.e. $tlf_p > T$, depreciation amounts to the difference between the purchase time in period t and the end of the planning horizon divided by the difference between the discontinuation time and period T. However, if the product is expected to be discontinued earlier than the end of the planning horizon, i.e., $tlf_p < T$, the depreciation rate will be the difference between the purchase time at period t and the discontinuation time.

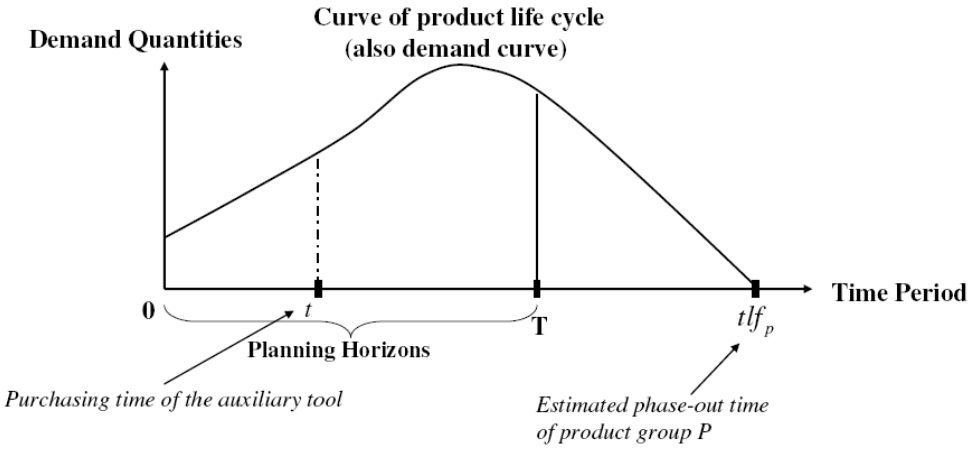


Figure 3. Linear depreciation based on the product life cycle

$$depreciation\_value = \frac{\min\left(T, tlf_p\right) - t + 1}{tlf_p - t + 1} \tag{1}$$

The following assumptions are considered to formulate the problem. (1) It is assumed that demand for each product in each period can be divided into different cycles and plant areas for production. (2) The production lead time of each plant is assumed to be negligible. (3) It is assumed that the supply of key materials will not be interrupted, hence, material shortage is not allowed. (4) Purchasing exclusive accessory tools, masks for the bottleneck machines are assumed to be sufficient for expanding the production capacity of each plant. Finally, (5) the model only considers the production process of arrays and does neither consider the possible bottleneck in the assembly process nor the material-related characteristics. The following notations are defined to formulate the problem.

Indices and parameters

i	Manufacturing plant, $i = 1,2,...,I$
k	Product family, $k = 1,2,...,K$
t	Time index, $t = 1,2,...,T$
de_{kt}^{ξ}	Demand for product family k in period t (piece)
pr_{kt}	The profit margin of selling one unit of product k in period t (USD)
ph_k	Phase-out (discontinuation) time of product family k
ca_{ikt}	Plant i's initial capacity for production of product family k in period t (sheet)
cw_{it}	Plant i's initial total capacity in period t (sheet)
cf_{ik}	Capacity consumption rate at plant i to produce product k (per sheet)
cr_{ik}	Production cut-rate at plant i to produce product k (pieces per sheet)
ye_{ikt}	Production yield rate at plant i for product k in period t (per sheet)
ec_{ikt}	Cost of purchasing one unit of the new auxiliary tool to produce product k at plant i in period t
eb_{ik}	The upper limit for the expansion of the overall production capacity of product k at plant i (sheet)
eu_{ik}	Increased capacities to purchasing one auxiliary tool of product k at plant i sheet)
ea_{ik}	Binary parameter for the expansion possibility for product k at plant i
ξ_{kt}	Demand state for product family k in period t
tm_{ikt}	Cumulative purchasing amount (capacity state) of the new auxiliary tool of each product k at plant i until period t
S_t	State variable in period t
A_t	Action taken in period t
N	Number of demand trajectories
M	Number of expansion trajectories

Decision variables

EM_{ikt} Integer decision variable for capacity expansion; the number of added auxiliary tools at plant i for product k in period t. The overall value in period t is $EM_t = \left(EM_{i\in I,k\in K,t}\right)_{|I|\times|K|}$

XQ_{ikt}: Continuous decision variable for capacity allocation; the amount of production quantity at plant i for product k in period t. The overall value in period t is $XQ_t = \left(XQ_{i\in I,k\in K,t}\right)_{|I|\times|K|}$ (sheet)

2.2. State, action, and state transition of the SDP model

Given the interdependence between early and late demand states along the planning horizon, a stochastic model with finite Markovian demands is used to consider the possible influence on capacity decisions. That is, the decision-making process considers the demand state and planning results of the previous period to determine the current decision. This procedure, named observe-decide-observe-decide, continues with every progression to the next planning period. This feature conforms to the concept of inter-related decisions, the so-call sequential effect (Pourhejazy et al. 2020), in dynamic programming and represents the probability of state transitions between consecutive stages.

State variables are classified into controllable and uncontrollable variables. The capacity planning problem decides a possible capacity expansion; hence, it is a controllable variable. Demand state is an uncontrollable variable that is determined considering the transition probability using Markov Chain. The state variable in period t is defined by $S_t = \left(tm_t, \xi_t \right)$. Given ξ_{kt} and tm_{ikt} representing the demand state and the number of available auxiliary tools for the production of product k in period t, tm_t, ξ_t are calculated using Equations (2) -(3), respectively. It should be noted that ξ_{kt} accepts three values $\xi_{kt} \in \{H, M, L\}$.

$$tm_t = \left(tm_{i\in I, k\in K, t} \right)_{|I|\times|K|} = \begin{pmatrix} tm_{11t}, tm_{12t}, ..., tm_{1Kt}, \\ tm_{21t}, tm_{22t}, ..., tm_{2Kt}, \\ ..., \\ tm_{I1t}, tm_{I2t}, ..., tm_{IKt} \end{pmatrix}, \; tm_{ikt} \in \left\{ 0,1,2,..., \min\left(\left\lfloor \frac{eb_{ikt}}{eu_{ik}} \right\rfloor, \; M \times ea_{ik} \right) \right\} \tag{2}$$

$$\xi_t = \left(\xi_{k\in K, t} \right)_{|K|\times 1} = \left(\xi_{1t}, \xi_{2t}, ..., \xi_{Kt} \right) \tag{3}$$

To define the actions, i.e., the capacity expansion decisions, the array representing the number of auxiliary tools and the amount of input in period t are defined using Equations (4) - (5), respectively.

$$EM_t = \left(EM_{i\in I, k\in K, t} \right)_{|I|\times|K|} = \begin{pmatrix} EM_{11t}, EM_{12t}, ..., EM_{1Kt}, \\ EM_{21t}, EM_{22t}, ..., EM_{2Kt}, \\ ..., \\ EM_{I1t}, EM_{I2t}, ..., EM_{IKt} \end{pmatrix}, \; EM_{ikt} \in \left\{ 0,1,2,..., \min\left(\left\lfloor \frac{eb_{ikt}}{eu_{ik}} \right\rfloor - tm_{ikt}, \; M \times ea_{ik} \right) \right\} \tag{4}$$

$$XQ_t = \left(XQ_{i\in I, k\in K, t} \right)_{|I|\times|K|} = \begin{pmatrix} XQ_{11t}, XQ_{12t}, ..., XQ_{1Kt}, \\ XQ_{21t}, XQ_{22t}, ..., XQ_{2Kt}, \\ ..., \\ XQ_{I1t}, XQ_{I2t}, ..., XQ_{IKt} \end{pmatrix} \tag{5}$$

The state transition probability is defined based on Equations (6)-(8); it represents the transition probability to period $t+1$ after a decision is made in period t . It is assumed that the capacity expansion and demand states are independent of each other, hence, Equation (5) can be obtained by simply multiplying Equations (6) and (7).

$$\begin{aligned} P\left(S_{t+1} \mid S_t,\ EM_t\right) &= P(S_{t+1}=\left(tm_{t+1},\ \xi_{t+1}\right) \mid S_t=\left(tm_t,\ \xi_t\right),\ EM_t) \\ &= P(tm_{t+1},\ \xi_{t+1} \mid tm_t,\ \xi_t\ ,\ EM_t) \\ &= P\left(tm_{t+1} \mid tm_t,\ EM_t\right) \times P\left(\xi_{t+1} \mid \xi_t\right) \end{aligned} \tag{6}$$

$$\begin{aligned} &P\left(tm_{t+1} \mid tm_t,\ EM_t\right) \\ &= P\left((tm_{11(t+1)}, tm_{12(t+1)}, ..., tm_{IK(t+1)}) \mid (tm_{11t}, tm_{12t}, ..., tm_{IKt}),\ (EM_{11t}, EM_{12t}, ..., EM_{IKt})\right) \\ &= \prod_i \prod_k P\left(tm_{ik(t+1)} \mid tm_{ikt},\ EM_{ikt}\right) \end{aligned} \tag{7}$$

$$\begin{aligned} P\left(\xi_{t+1} \mid \xi_t\right) &= P\left(\left(\xi_{1(t+1)}, \xi_{2(t+1)}, ..., \xi_{K(t+1)}\right) \mid \left(\xi_{1t}, \xi_{2t}, ..., \xi_{Kt}\right)\right) \\ &= \prod_k P\left(\xi_{k(t+1)} \mid \xi_{kt}\right) \end{aligned} \tag{8}$$

2.3. Reward function and optimality equation of the SDP model

Given dynamic programming equations provided in Section 2.2., a deterministic embedded Linear Programming (LP) model is developed to calculate one-period immediate reward function by Expressions (9) – (15). Equation (9) is the reward function that estimates the acquired profit $P_t(tm_t, \xi_t)$ from the capacity allocation plan after considering the occurring expansion cost $C_t(EM_t)$ when action A_t is taken in state S_t. Equation (10) calculates the capacity expansion cost $C_t(EM_t)$ considering the planning period when the product family is discontinued from the company's portfolio and the time the auxiliary tool is purchased. Expressions (11) – (15) forms the LP model of the capacity allocation model that seeks profit maximization $P_t(tm_t, \xi_t)$, as shown in Equation (11). Constraints (12) define the demand-fulfillment condition. Constraints (13) set a restriction on the total production capacity of each plant in period *t*. Constraints (14) determine the available capacity for each product family at each plant in period *t*. The available capacity of each product family of each plant is defined as the original capacity ca_{ikt} plus the cumulative expansion capacity $tm_{ikt} \times eu_{ik}$. Constraints (15) indicate that the production variable accepts non-negative values.

$$R_t(S_t,\ EM_t) = P_t(S_t) - C_t(EM_t) = P_t(tm_t, \xi_t) - C_t(EM_t) \tag{9}$$

$$C_t(EM_t) = \sum_i \sum_k \left(\left(EM_{ikt} \times ec_{ikt}\right) \times \left(\frac{\min\left(ph_k, T\right) - t + 1}{ph_k - t + 1} \right) \right) \tag{10}$$

$$P_t(tm_t, \xi_t) = Max\left\{\sum_i \sum_k \left(pr_{ikt} \times XQ_{ikt} \times cr_{ik} \times ye_{ikt}\right)\right\} \tag{11}$$

Subject to:

$$\sum_i \left(XQ_{ikt} \times cr_{ik} \times ye_{ikt}\right) \le de_{kt}^{\xi} \quad \forall k \in K \tag{12}$$

$$\sum_k \left(XQ_{ikt} \times cf_{ik}\right) \le cw_{it} \quad \forall i \in I \tag{13}$$

$$XQ_{ikt} \le ca_{ikt} + tm_{ikt} \times eu_{ik} \quad \forall i \in I,\ \forall k \in K \tag{14}$$

$$XQ_{ikt} \ge 0 \quad \forall i \in I,\ \forall k \in K \tag{15}$$

Equation (16) is the finite-horizon optimality recursive equation that determines the best capacity expansion and allocation decisions in each planning period to maximize net profit. In this equation, $V_t^*(S_t)$ represents the maximum net profit in state S_t after applying EM_{ikt}. This value is referred to as the immediate reward that can be obtained in period t when compared with period $t+1$. In this equation, $V_T^*(S_T)$ represents the boundary condition in period T that is equal to the current net profit value $P_T(tm_T, \xi_T)$, that is no decisions will be made until period T, $V_T^*(S_T) = P_T(S_T) = P_T(tm_T, \xi_T)$.

$$\begin{aligned} V_t^*(S_t) &= \max_{EM_t}\left\{R_t(S_t,\ EM_t) + E\left[V_{t+1}^*(S_{t+1})\right]\right\} \\ &= \max_{EM_t}\left\{R_t(S_t,\ EM_t) + \sum_{S_{t+1}} P(S_{t+1} \mid S_t,\ EM_{ikt}) \times V_{t+1}^*(S_{t+1})\right\} \end{aligned} \tag{16}$$

Given the defined reward function and the optimality equation, the computational complexity evaluated by the backward induction algorithm is provided as Equation (17). The computational complexity of the optimality equation increases when more production plants (I), products (K), length of the planning horizon (T) are involved. By S representing the number of demand states, $ub_{ik} = \left\lfloor \frac{eb_{ik}}{eu_{ik}} \right\rfloor$ representing the maximum number of auxiliary tools for each product family at each plant, and assuming that all production plants have the potential of expanding capacity for all product families, $ea_{ik} = 1\ \forall i,k$,

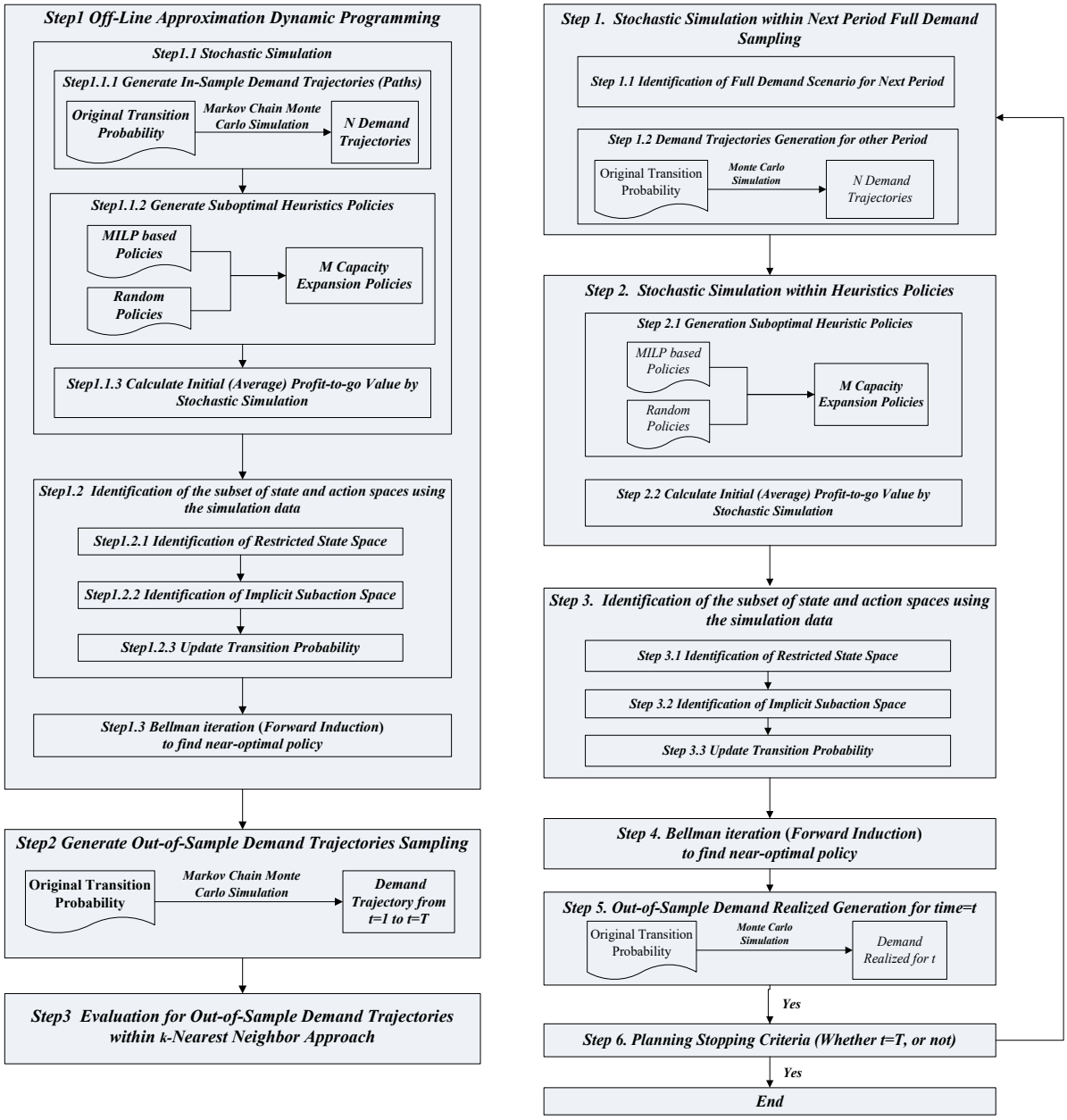
$$f_{complexity} = (T-1) \times S \times \sum_{i=0}^{ub_{11}} \sum_{j=0}^{ub_{12}} \cdots \sum_{k=0}^{ub_{I(K-1)}} \sum_{l=0}^{ub_{IK}} (i+1) \times (j+1) \times \ldots \times (k+1) \times (l+1) \tag{17}$$

Considering $T = 2$, $K = 2$, $I = 2$, $S = 2$, $ub_{ik} = 2\ \forall i,k$, the number of function evaluations amounts to 2592. The computational complexity will be less if there is a restricted limitation on

the production capacity of a product family at a certain manufacturing plant, i.e., if capacity cannot be expanded. Two dynamic programming algorithms are developed in the next subsections to solve the problem.

2.3. k-Nearest-Neighbor-based ADP

For stochastic dynamic capacity planning with many factories, products, and long planning periods, insufficient sampling and ignoring the states with a lower likelihood of occurrence may result in poor judgment based on the obtained solution. We propose to integrate the *k*-Nearest Neighbor method into the Simulation-based ADP to address this shortcoming. *k*-ADP consists of three major computational steps; First, the off-line ADP takes place. Next, the out-of-sample demand trajectories are generated. Finally, the computation is completed by evaluating the out-of-sample demand trajectories using the *k*-nearest neighbor method. We now elaborate on the computational procedure shown in Figure 4(a).



(a) *k*-ADP (b) *R*-ADP

Figure 4. The computational procedure of the proposed methods

2.3.1. Off-line approximation and the simulation procedure

This phase consists of applying random simulations guided by a heuristic, identifying the constrained state and sub-action spaces using the resulting simulation data, and forward induction via Bellman Iteration.

The procedure begins with stochastic simulation-based optimization that generates in-sample demand trajectories, finds the suboptimal states (decisions), and calculates the performance value of the resulting states. To begin with the simulation procedure, different parameter values are generated randomly using Markov Chain to represent unique situations in each period. The stochastic simulation then helps select the promising states to control the size of the state space. In the resulting sub-space, the optimizer finds the state with the best decision path. This procedure provides the Bellman Iteration procedure with the initial Profit-to-Go value, $V(S_t)$. In this procedure, applying various trajectories may result in the same state with different performance values. To address this situation, the average of the performance values of different state trajectories through the state is considered as the initial value. The initial value will be then considered in the procedure of finding the best decision trajectory.

First, in-sample demand trajectories should be generated. The stochastic simulation model in this study considers two separate modules corresponding to the demand and capacity states. MCMCS is used to carry out the demand transition process, the so-called in-sample demand trajectory. Given $j = 1,...,N$ demand sample paths obtained by MCMCS, a random number $Uniform(0,1)$ is used to generate the Demand Transition Matrix (DTM) of probabilities and cumulative distribution function, shown in Expressions (18) and (19), respectively, to estimate a new demand state.

$$DTM = \begin{array}{c} \\ L \\ M \\ H \end{array} \begin{array}{c} \begin{array}{ccc} L & M & H \end{array} \\ \begin{bmatrix} P_{LL} & P_{LM} & P_{LH} \\ P_{ML} & P_{MM} & P_{MH} \\ P_{HL} & P_{HM} & P_{HH} \end{bmatrix} \end{array} \tag{18}$$

$$\begin{array}{c} \\ L \\ M \\ H \end{array} \begin{array}{c} \begin{array}{ccc} L & M & H \end{array} \\ \begin{bmatrix} P_{LL} & (P_{LM} + P_{LL}) & 1 \\ P_{ML} & (P_{ML} + P_{MM}) & 1 \\ P_{HL} & (P_{HL} + P_{HM}) & 1 \end{bmatrix} \end{array} \tag{19}$$

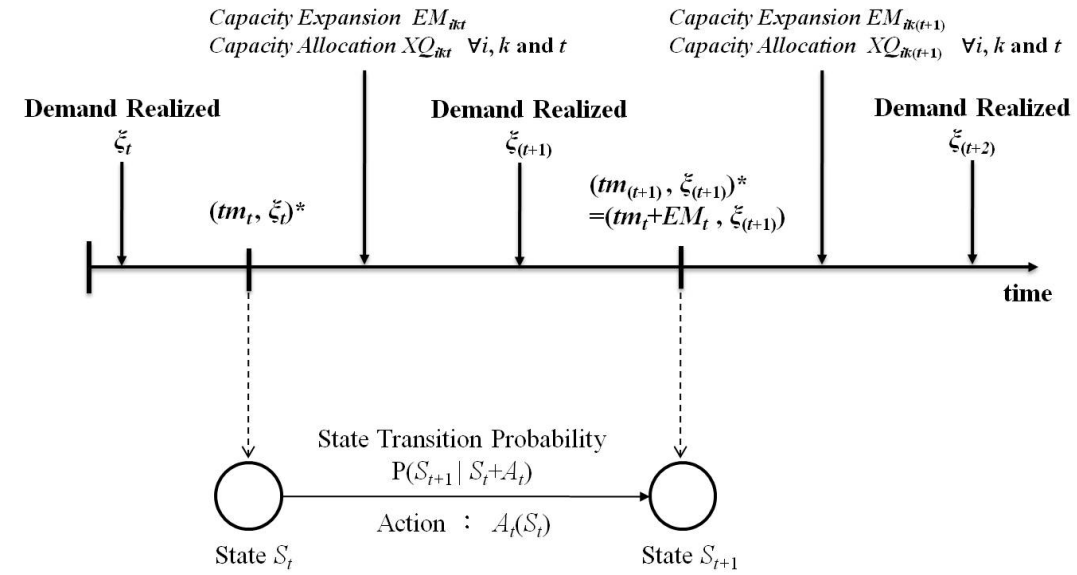
For example, let assume a high demand situation in the current period and the transition probabilities of 0.5, 0.3, and 0.2 to high, medium, and low in the next period, respectively. Given the cumulative vector of probabilities, $[0.5 \quad 0.8 \quad 0.1]$, if the randomly generated number is less than 0.5, the demand status in the next period remains high. Else, if the random number is between 0.5 and 0.8, the next period's demand declines to medium and if it is between 0.8 and 1, demand in the new period will be low. The pseudocode of this procedure is provided in Appendix A.

Next, suboptimal search policies are defined. Given the resulting demand trajectory, the optimum capacity expansion decision, $EM_{ik}(t)$, can be determined by solving the LP formulation presented in section 2.1. For complex decision environments, the outcomes of deterministic mathematical programming are not robust and various values for the uncertain factors should be considered to find the best course of action. The set of resulting optimal decisions is regarded as the action pool. This research proposes to consider random actions in addition to the LP outcomes to diversify the results. For this purpose, the capacity expansion restriction in Equation (4) is considered to find random feasible decisions. Given the maximum capacity of plant i for manufacturing product k, $eb_{ik} = 0$, and the unit capacity for production of product k at plant i, eu_{ik}, the expansion upper limit is $\frac{eb_{ik}}{eu_{ik}}$, the non-integer part will be rounded-up if larger-or-equal-to 0.5 and rounded-down if it is less than 0.5. Equation (20) is considered for generating random capacity expansion decisions. In this equation, $ea_{ik} = 1$ if plant i can add the auxiliary tool for manufacturing product k and $ea_{ik} = 0$, otherwise. The upper limit expansion value is deducted from the randomly generated first-period expansion decisions to estimate the new expansion upper limit; this procedure continues until the last planning period, T.

$$0 \le EM_{ikt} \le \min\left(\left\lfloor \frac{eb_{ik}}{eu_{ik}} \right\rfloor - \sum_{t'=1}^{t} EM_{ikt'}, M \times ea_{ik}\right) \quad \forall i \in I, \forall k \in K, \forall t \in T \tag{20}$$

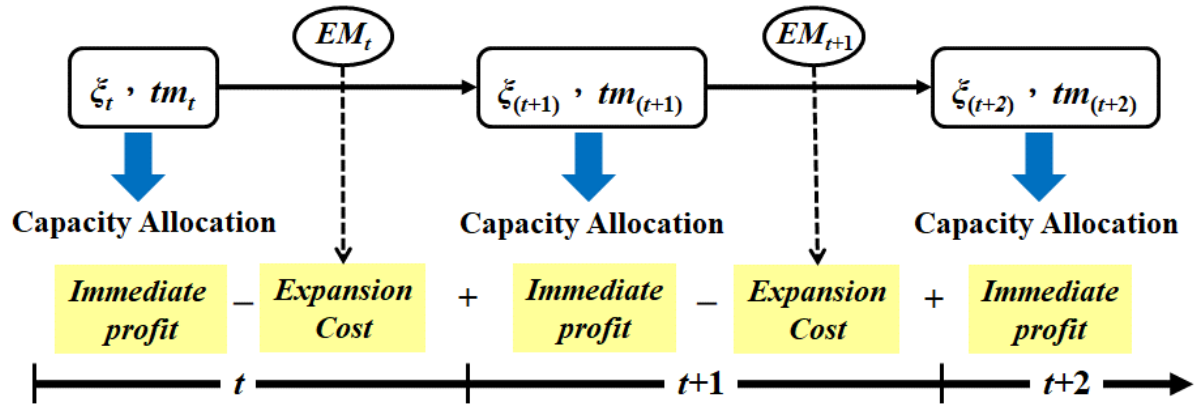
Finally, the Profit-to-Go value should be measured. In the simulation procedure, the total number of state trajectories is $N \times M$. The Profit-to-Go value of the resulting $N \times M$ states can be calculated using Equation (21). The above procedure continues until period T, as illustrated in Figure 5. In this approach, $R_t(S_t,\ EM_t)$ represents the revenue in period t considering the demand scenario and the cumulative expansion of the capacity state calculated using Equation (7).

$$V(S_t)\{\textit{total future expected profit}\} \cong \mathrm{V}^0(\mathrm{S_t}) = \sum_{i=0}^{\mathrm{T}} R_t(S_t,\ EM_t) \tag{21}$$



$^*(tm_t, \xi_t)$ = (Matrix of total purchasing amount of the new auxiliary tool until period t, $tm_t = \left(tm_{i\in I,k\in K,t}\right)_{|I|\times|K|}$,
Vector of demand state in period t, $\xi_t = \left(\xi_{k\in K,t}\right)_{|K|\times 1}$)

(a). Timing diagram



(b). Computation procedure

Figure 5. Schematic illustration of the studied dynamic capacity planning

In this procedure, the performance value may vary from one state to another if different expansion decisions are allowed. If the capacity expansion state transferred to the next period remains the same, the performance value can be different due to a possible demand state transition in the next period. Considering that a particular state can be repeated n times, the average profit, $V(S_t) = \frac{\sum V(S_t)^n}{n}$, will be considered to represent the initial performance value, which will be used as input to the Bellman Iteration procedure.

To identify the states and actions subsets, simulation data should be used. Given various demand scenarios and capacity expansion decisions, one should categorize the obtained simulation data into different subsets, as shown in Figure 6, to reduce the space size and the computational time. For this purpose, the scope of the state space selected for simulation data should be first determined. Next, the capacity expansion decision space should be selected

applying a heuristic policy. Finally, the transition probability defining the Markov Chain procedure should be updated.

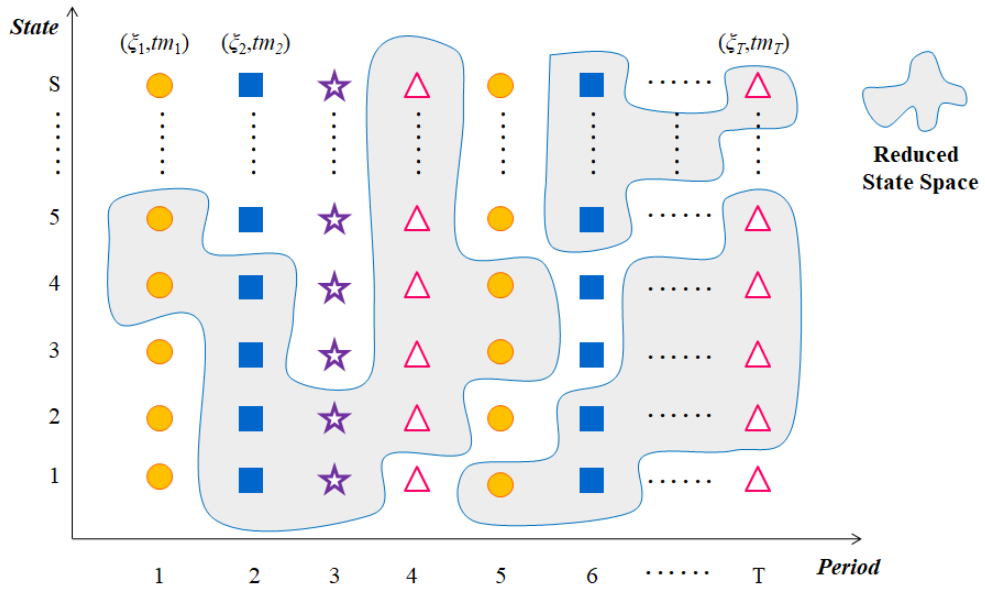
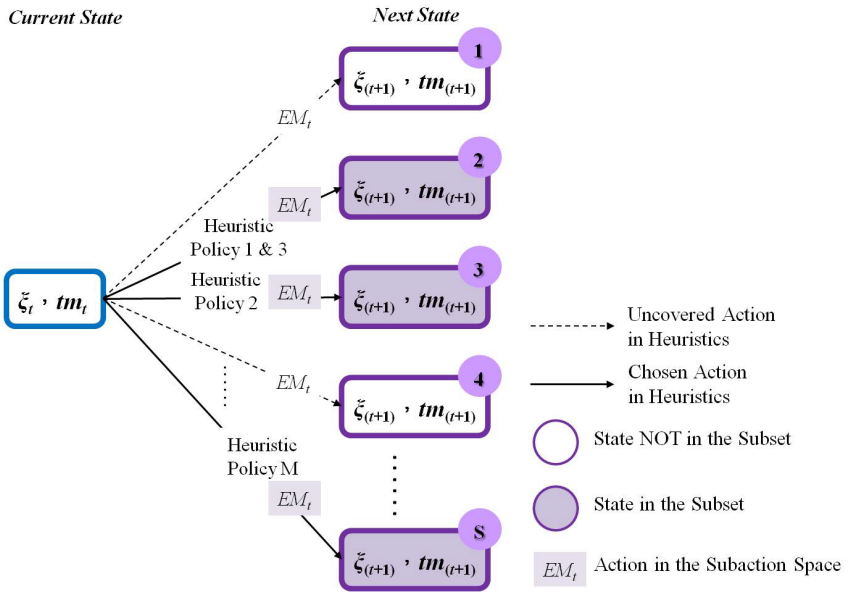
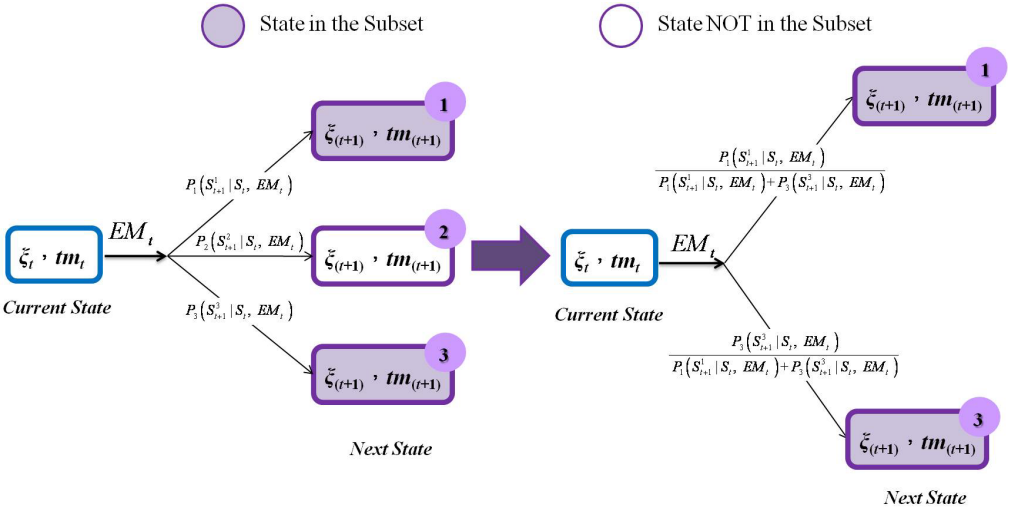


Figure 6. Categorization of the state space into subsets in the reduced state space

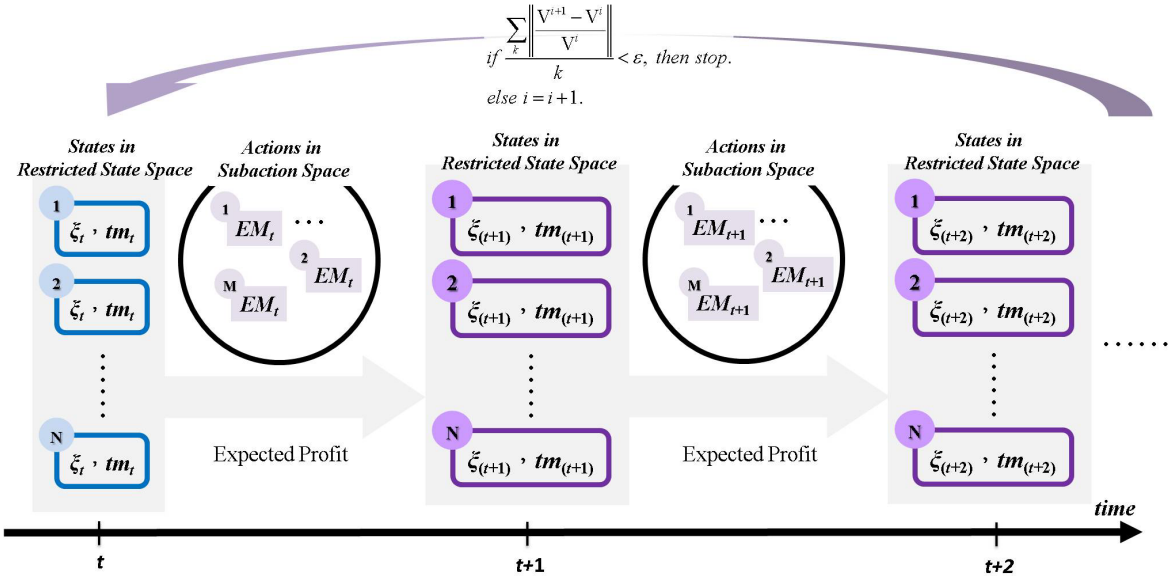
From the resulting $N \times M$ state trajectories, the states associated with a very small transfer probability value and those resulting in poor capacity expansion decisions should be filtered from the simulation procedure. Besides, the states that occur multiple times should be recorded once after integrating them. Theoretically, sufficient simulation runs are required to evaluate all possible states to find the best. However, it is not feasible to enumerate all states when the state space of the problem is large. Therefore, an approximation strategy is needed to ignore the states with an unlikely transition probability. For this purpose, the Bellman Iteration is applied, which requires the state transition probability in each period; the transition probability value for the states in the reduced state space must be updated. To reduce the computational complexity of Bellman Iteration, each state only considers the decisions in the sub-action space, $U^{X(k)}$, such that the number of decision-making sub-sets is equal to the number of states in the finite state set, as shown in Figure 7(a). Heuristic i (h_i) will be selected when state $X(k)$ applies its capacity expansion decision. The resulting set is presented by $\mathrm{A}^{S_t} = \{a_1, ..., a_i\}$ where $a_i = h_i[S_t], \forall i \in \{1, ..., M \times N\}$.



(a). Initiating the transition diagram.



(b). Updating the transition probability after each transition.



(c). Bellman iteration procedure.

Figure 7. State transition computational procedure in *k*-ADP

Figure 7(b) illustrates the procedure of updating the transition probability value for the states in the reduced state space. In the illustrative example, the capacity decision in the original state (S_t) may be transferred to three states ($S_{t+1}^1, S_{t+1}^2, S_{t+1}^3$) with the corresponding transition probabilities $P_1\left(S_{t+1}^1 \middle| S_t, EM_t\right)$, $P_2\left(S_{t+1}^2 \middle| S_t, EM_t\right)$, and $P_3\left(S_{t+1}^3 \middle| S_t, EM_t\right)$, respectively. Since S_{t+1}^2 is not included in the reduced state space, it is necessary to recalculate the transition probability of EM_t in state S_t to the next state. For this purpose, the transfer probability of S_{t+1}^1 and S_{t+1}^3 should be updated to $\frac{P_1(S_{t+1}^{(1)}|S_t, EM_t)}{P_1(S_{t+1}^{(1)}|S_t, EM_t) + P_1(S_{t+1}^{(3)}|S_t, EM_t)}$ and $\frac{P_1(S_{t+1}^{(3)}|S_t, EM_t)}{P_1(S_{t+1}^{(1)}|S_t, EM_t) + P_1(S_{t+1}^{(3)}|S_t, EM_t)}$, respectively.

As a final procedure in Step 1, Bellman Iteration is used to calculate the optimal Profit-to-Go function, $V\left(S_t\right)$. The Forward Induction method helps update the performance value in each state, which is different from the traditional dynamic programming that starts from the last period in Backward Induction. Bellman Iteration only calculates the Profit-to-Go value of the states included in the reduced state space. That is, the decision subspace in A^{S_t} is identified from the set of best decisions instead of the entire decision space. The expected performance value of each state decision is finally calculated considering the probabilities resulting. Equation (22) is used to find the best decision considering various conditions in each period. The Bellman Iteration procedure continues until V^i reaches a convergent state and the procedure terminates when the condition in Equation (23) is met. This procedure is visualized in Figure 7(c).

$$V^{i+1}(S_t) = \max_{EM_t \in A^{S_t}} E\left\{R[S_t, EM_t] + V^i[S_{t+1} \mid S_t, EM_t]\right\} \tag{22}$$

$$\frac{\sum_k \left\| \frac{V^{i+1} - V^i}{V^i} \right\|}{k} < \varepsilon \tag{23}$$

2.3.2. Generating out-of-sample demand trajectories

Given the state and decision sub-sets defined and the best decision identified by the Bellman Iteration, MCMCS is applied to sample the demand path with the exception that the number of samples this time is greater than the number of samples in Step 1. The purpose is to investigate whether the best solution obtained through Step 1 is the same considering other situations, i.e., demand scenarios.

2.3.3. Evaluating out-of-sample demand trajectories using the k-Nearest Neighbor method

The Bellman Iteration method was applied to determine the best decision from the states that are present in the reduced state space in Step 1. However, it is not certain to know if better decisions can be identified from the non-present states. This study proposes to apply the *k*-Nearest Neighbor Approximation considering a two-stage method to address this shortcoming.

k neighboring states of the existing states, which have similar characteristics, are first identified to help estimate the state(s) that may occur but has no decision information. Considering a distance parameter, δ, Equation (24) determines the area within which these neighbor states are situated.

$$k = \left\{ s_{in} \in S_{sub} : d = \sqrt{(s_{in} - s_{out})^T (s_{in} - s_{out})} < \delta \right\} \quad (24)$$

In this equation, S_{in} is the demand state from the set of existing states and S_{out} represents the demand state outside the reduced set of the states. Let consider an illustrative example of 2 plants, 2 products, and two possible states of High-demand (H) and Low-demand (L). Assuming the initial state of $(0,0,0,0)$, the transitions and their corresponding decision in the reduced state space are

No.	State				Action			
1	0	0	0	0	0	0	0	0
					0	0	0	1
2	0	0	1	0	1	0	0	0
3	1	0	0	1	0	0	0	0
4	1	0	1	0	0	1	0	0
5	1	1	1	0	0	0	0	1
6	1	1	1	1	0	0	0	0

, where the actions associated with the demand state (0,0,0,0) are (0,0,0,0) and (0,0,0,1) and the other states correspond to only one action. $\delta = 1.5$ is considered to search for the neighbor states considering the existing state. Assuming a high demand for the current period, the initial decision can be regarded as $(H,0,0,0,0)$. In this situation, $k = 4$ results in the following set of neighbor states, $\{(0,0,0,0),(0,0,1,0),(1,0,0,1),(1,0,1,0)\}$. There may be situations in which no neighboring states can be found considering distance δ, hence, the decision remains unchanged. In doing so, accumulating the performance value of the selected state in period *t* to the last period and reducing the corresponding cost results in the expected net profit value under the current demand trajectory.

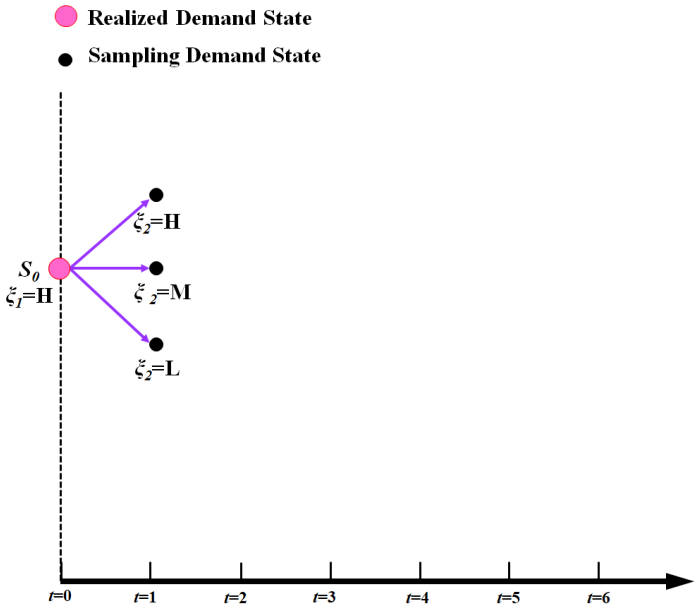
The next step is to evaluate the identified sub-action, i.e., the states that do not belong to the reduced state space under the current demand scenario, S_{out}. Assuming that the decision is to expand the capacity in the current period, $R\left[S_{t+1} \middle| S_t, EM_t\right]$ represents the profit obtained after capacity allocation in the next immediate period and $C\left[S_t, EM_t\right]$ is the capacity expansion cost. In this stage, the greatest profit can be identified using Equation (25).

$$A^{Best}(S_t) = Arg \max_{EM_t \square A^{S_t}} E\left\{C\left[S_t, EM_t\right] + R\left[S_{t+1} \mid S_t, EM_t\right]\right\} \quad (25)$$

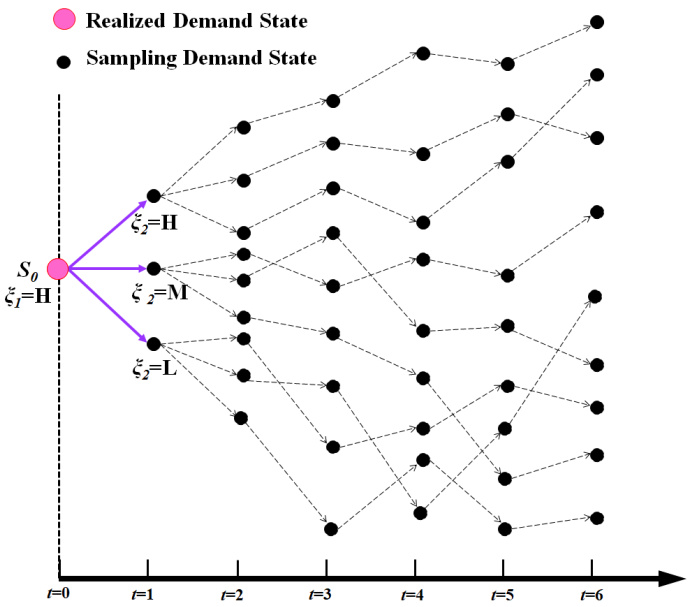
2.4. Rolling-Horizon-based Approximate Dynamic Programming

As stated earlier in the section, insufficient sampling in approximate dynamic capacity planning for solving real-world problems may result in a poor estimation of the best decision, sub-optimal solutions. In this section, we propose the concept of Next-Period-Full-Demand-Sampling (BPFDS) based on which the *R*-ADP method is developed. R-ADP with Next-Period-Full-Demand sampling combines both in-sample and out-of-sample (realized demand) to find solutions for robust capacity expansion and allocation decisions on a rolling-horizon basis. In this concept, demand appears as the state of measure in planning once it is realized. To reflect on this concept, at period *t*, all possible demand states are considered in the calculations when

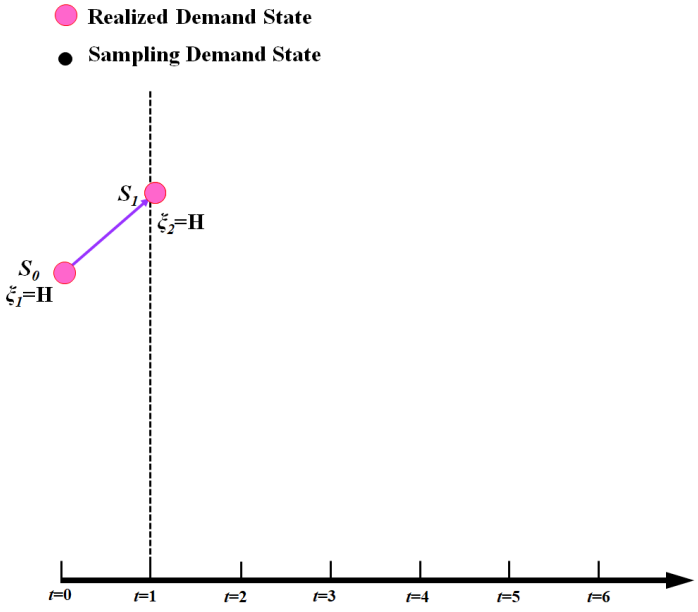
the time is one period ($t+1$) away from the current period while the in-sample demand state in the remainder of the periods (i.e., after $t+2$) is determined using MCMCS sampling. Then, once the period is transferred from t to $t+1$ and a specific out-of-sample demand scenario is realized in period $t+1$, other demand scenarios that did not come to realization will no longer be considered in the new calculations. This approach not only reduces the state space but also ensures the validity of decisions in each period. The concept of Next-Period-Full-Demand-Sampling is illustrated in Figure 8.



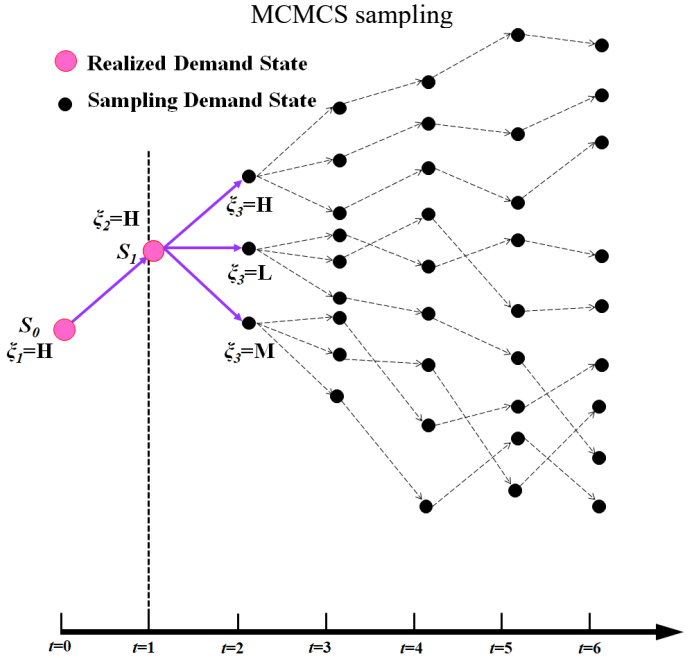
(1) All possible demand states are considered in period t=1



(2) In-sample demand trajectory for the remaining periods using MCMCS sampling



(3) Move to t=1 and a specific out-of-sample demand scenario **H** is realized



(4) All possible demand states are considered in period t=2 and in-sample demand trajectory for the remaining periods using MCMCS sampling

Figure 8. Demand transition procedure of *R*-ADP in the rolling-horizon basis

In this example, it is assumed that the state of realized demand in $t = 1$ is High, hence, all possible demand scenarios, i.e., $\{L, M, H\}$ are considered for $t = 2$. Once the period is transferred from $t = 1$ to $t = 2$, the demand scenarios that did not occur are not considered in the new calculations. As shown in the figure, the demand state sampling from $t = 3$ to $t = 6$ is performed considering the possible scenarios and probabilities. The best decision under the high demand in $t = 1$ can be obtained by measuring the performance value of each demand trajectory and selecting the best state. *R*-ADP consists of six major computational steps shown in Figure 4(b).

2.4.1. Stochastic simulations applying the concept of Next-Period-Full-Demand-Sampling

First, the set of demand states in the period is defined. Considering the demand state as an uncontrollable variable, it is not possible to be certain about a particular scenario in the next period. To ensure the robustness of planning, exhausting all the possible scenarios in period $t+1$ is considered. In our example, three scenarios $\{L, M, H\}$ will be considered to determine the demand state of the next period. Then, a demand trajectory for the remaining periods is generated. MCMCS is applied for sampling *N* demand trajectories in periods $t+2$ until $t = T$ based on the current situation, i.e., the demand state of period t. In this procedure, the states with a low probability of occurrence will not be included.

2.4.2. Stochastic simulation applying heuristics policies

The next step is to generate local heuristics policies. For this purpose, an LP-based relaxation policy and a policy based on selecting random actions are considered. This procedure is similar to that of *k*-ADP (Step 1.1.2). Then, the Profit-to-Go value should be calculated by applying the procedure detailed in Step 1.1.3 of *k*-ADP.

2.4.3. Identifying the reduced state and sub-action spaces

Applying a similar procedure to that of *k*-ADP (Step 1.2), Step 3 of *R*-ADP consists of (1) defining the scope of the reduced state space for simulations; (2) determining the capacity expansion decision space selected by the heuristic policy, and (3) updating the transfer probability values.

2.4.4. Applying Bellman Iteration

The Bellman Iteration procedure explained in subsection 2.3.3 is applied. In this procedure, the Profit-to-Go value of the finite state set resulting from Step 3 will be calculated. This process is limited to the sub-action space, $U^{X(k)}$, to search among the best set of decisions instead of enumerating all the possible decisions. In doing so, Bellman Iteration finds the best decision across different situations in each period using Equation 28.

2.4.5. Generating the realized demand (out-of-sample) for the new period

MCMCS is applied for sampling the demand scenarios for $t+1$. Random numbers generated using $Uniform(0,1)$ are used to determine the cumulative distribution function for determining the demand transition probability.

2.4.6. Evaluating the termination criterion.

If $t < T$, proceed to the next period ($t = t + 1$) and repeat the computational procedure explained in subsection 2.4.1. Otherwise, i.e., $t = T$, the termination condition is met, and the algorithm process should be terminated.

3. Numerical analysis

This section considers instances of various configurations to evaluate the performance of *k*-ADP and *R*-ADP comparing them with the SDP approach as the baseline algorithm. All the experiments are conducted using C programming language and CPLEX on a personal computer with the following specs: Intel® Core (TM) i7-6700 CPU 2.8GHz, 4 GB of RAM, and Windows 10 operating system. The numerical analysis begins with explaining the data set and the performance measures used for benchmarking the algorithms; continues with the analysis of the results and concludes the section with statistical analysis to confirm the findings.

3.1. Performance measures

The performance measures considered in this study are categorized into in-sample and out-of-sample to evaluate the quality of the solutions. The in-sample measures consider Optimum Profit (OP), Performance Gap (PG), Explored Space (ES), and the computational time (CPU). The out-of-sample measures take into account the Average Profit (AP), Performance Gap (PG), Standard Deviation (StD), and Policy Difference (PD). Profit is considered as the main operational measure to compare the resulting financial outcomes from the dynamic programming approaches. AP is the average profit value over 100 sets of demand scenarios, which are estimated using Monte Carlo simulations. PG determines the relative difference between the best results obtained by SDP with that of *k*-ADP and *R*-ADP, which can be calculated using Equation (26) where P_{ADP} demonstrated the optimum profit obtained by the developed *k*-ADP and *R*-ADP algorithms. The optimum solutions may recommend various strategies, i.e., through capacity expansion, to meet the market demand and maximize profit. The mean square of the difference between *k*-ADP and SDP strategies, $PD = \sqrt{\sum_{d=1:Dimension} \left(S_d^{kNNADP} - S_d^{SDP}\right)^2}$, is considered to analyze the trade-off between various optimal solutions and evaluate their quality.

$$PG = \frac{P_{ADP} - P_{SDP}}{P_{SDP}} \tag{26}$$

From a computational perspective, ES determines the effectiveness of the algorithm considering the total number of states it has searched to arrive at the result. Besides, CPU time is used to compare the efficiency of the solution algorithms. Given the stochasticity of the results obtained by *k*-ADP and *R*-ADP and the dependence of the procedure on the simulation outcomes, 100 independent runs are considered for out-of-sample experiments to analyze the results. For this purpose, average (Ave), minimum (Min), and maximum (Max) values determine the average, worst, and best outcomes obtained in the random process, respectively.

3.2. Results analysis

Five instances are considered to evaluate the performance of the developed algorithms against the baseline algorithm, SDP. Specifications of the test instances are summarized in Table 1. First, the outcomes of an illustrative small instance are analyzed in detail. A real-world example comes next to evaluate the models' practicability. Finally, three large-scale exemplary instances are presented to shed light on the computational capabilities of the developed dynamic programming approaches in real-time capacity planning for industrial applications. The model parameters summarized in Table 2 are considered to analyze the results and discuss their implications.

Table 1. Specification of the test instances.

Instance	Setting (f, p, t)	Demand Over Periods	Demand State	Capacity Variable	Capacity State	Total no. of states	No. of actions
1	2,2,4	Homogenous	2	4	16	96	16
2	2,5,6	Homogenous	3	10	7.87×10^{3}	1.18×10^{6}	7.87×10^{6}
3	2,5,6	Heterogenous	3	10	7.87×10^{3}	9.57×10^{7}	7.87×10^{6}
4	3,6,6	Homogenous	3	18	5.17×10^{8}	7.75×10^{9}	5.17×10^{8}
5	3,7,6	Homogenous	3	21	1.39×10^{10}	2.09×10^{11}	1.39×10^{10}

Table 2. Experimental factors considered to solve the capacity planning problem.

Instance	Factor		Level				
Small	Number of Demand Trajectory	NDT	2	6	10	14	18
	Number of Expansion Trajectory	NET	2	6	10	14	18
	Search Distance of *k*-Nearest Neighbor	δ	1	2	3	4	5
	The policy of Expansion Trajectory	PET	LPR	RND	Mix	-	-
Large	Number of Demand Trajectory	NDT	10	20	30	40	50
	Number of Expansion Trajectory	NET	10	30	50	70	90
	Search Distance of *k*-Nearest Neighbor	δ	1	2	3	4	5
	The policy of Expansion Trajectory	PET	LPR	RND	Mix	-	-

3.2.1. Illustrative instance

The first case consists of two array manufacturing plants each of which producing two product families, A and B. With the demand becoming known within the planning period, the capacity allocation and expansion decisions should be adjusted dynamically to maximize the net profit. The course of the planning period is three months, i.e., $T = \{0,1,2,3\}$. Two scenarios of more-than-average (High; H) and less-than-average (Low; L) demand intensity is considered for each period, where homogenous demand changes for the product families are assumed, that is $\xi_t \in \{H, L\}$, $\forall \mathrm{t} \in \mathrm{T}$. The state transition and the computational complexity are described below.

Assuming a high-demand situation for the current period, demand for the next period remains high with a probability of 0.7 and may degrade to low with a probability of 0.3. Alternatively, if the current period is demonstrating a low-demand situation, the next period's demand may remain low with a likelihood of 0.8 or change to high with a likelihood of 0.2. From the supply viewpoint, if plant 1 increases its capacity for producing product A during period t, the production capacity increases starting from the next period. It is assumed that the production capacity cannot decrease after a capacity expansion decision. It is also assumed that the production capacity of one product at one plant can be expanded only once, that is $EM_{ikt} \in \{0,1\}$ $\forall \mathrm{i} \in \mathrm{I}, \forall k \in K, \forall t \in T$. Given 4 capacity expansion situations for each of 2 plants and 2 product families, 2 demand change scenarios in each period, and 3 planning periods, a total

of $4\times2\times2\times2\times3=96$ states have resulted. The computational complexity of the instances can be calculated using Equation (17); this value for the illustrative small instance amounts to 486.

Rollout and Bellman's approaches are considered to solve this instance using *k*-ADP and *R*-ADP. Table 3 summarizes the results for in-sample and out-of-sample experiments considering the small-scale instance. It is observed that *k*-ADP, using the Bellman approach, obtained an optimal solution like that of SDP while searching a small fraction, about 13 percent, of the solution spaces; *k*-ADP required one-third of the computational time required by SDP. *k*-ADP using the Rollout approach also searches about 13 percent of the solution space and demonstrated a computational time of about two-thirds of SDP. Considering the out-of-sample instances, both *k*-ADP and *R*-ADP obtain the same results as SDP and represent no Policy Differences.

Table 3. Results of solving the illustrative instance (in-sample and out-of-sample).

Category	Method		OP	PG (%)	ES	% ES	CPU (sec)
In-sample	SDP	-	49568	0.00	96	100.00	3
	k-ADP	Bellman	49568	0.00	13	13.54	1
		Rollout	49894	0.66	13	13.54	1
Out-of-sample	Method		AP	PG (%)	StD		PD
	SDP	-	48,938	-	2,904		-
	k-ADP	Bellman	48,938	0.00	2,904		0.000
		Rollout	48,938	0.00	2,904		0.000
	R-ADP	Bellman	48,938	0.00	2,904		0.000
		Rollout	48,938	0.00	2,904		0.000

OP = Optimum Profit; EP = Average Profit; PG = Performance Gap; StD = Standard Deviation; PD = Policy Difference.

Table B1 in Appendix B provides a detailed analysis of in-sample and out-of-sample solutions obtained by *k*-ADP considering various demand and expansion trajectories. For the in-sample case, it is observed that with an increase in the demand trajectory, the Average Profit of the solution obtained by *k*-ADP approaches that of SDP, hence, the Solution Gap approaches zero. From a computational perspective, the changes in the number of searched states resulting from an increase in the number of demand trajectories depend on the applied search policy. For example, under the LP-relaxation policy, the number of searched states increases with an increase in the number of demand trajectories and gets converged after a certain point.

It is also observed that the Average Profit resulting from *k*-ADP approaches that of the best solution obtained by SDP when an increase in the number of capacity expansion trajectories is accompanied by a suitable search policy, i.e., LP-relaxation or mixed. Another notable observation is that applying the mixed policy, i.e., LP-relaxation and Random explores more states than applying LP-relaxation or Random policies. Overall, an increase in the number of capacity expansion trajectories increases the total number of searched states while under the LP-relaxation policy, this value remains the same. Considering the search approach, if LP-relaxation or mixed policies are used, *k*-ADP obtains the same optimal solution found by SDP. However, there is a gap between the optimal solutions obtained by *k*-ADP and that of SDP when the Random policy is applied; this gap is smaller when greater demand or capacity expansion trajectories are considered.

In out-of-sample experiments, the optimal solutions found by *k*-ADP using both Bellman and Rollout approaches have the same Average Profit as that of SDP. Regardless of the number of demand trajectories, *k*-ADP under LP-relaxation or Mixed policies obtains the same Average Profit as SDP. However, using a Random policy for sampling the capacity expansion trajectory results in a Performance Gap of 0.57 percent where the demand trajectory is increased to its highest. Besides, it is observed that using LP-relaxation or Mixed approaches results in no Policy Differences regardless of the number of demand trajectories, meaning that the capacity expansion decisions found by *k*-ADP have the same quality as that of SDP. The resulting capacity expansion decision appears to be different and less likely to be optimum when the Random approach for sampling the capacity expansion trajectories is applied.

Considering various capacity expansion trajectory numbers, *k*-ADP obtains solutions with an Average Profit equal to that of SDP in all the configurations under LP-relaxation and Mixed policies. In cases of applying a Random policy, the Performance Gap between *k*-ADP and SDP decreases to a negligible value as the number of capacity expansion trajectories increases. In terms of Policy Difference, the configurations under LP-relaxation and Mixed policies obtain solutions as good as the solutions obtained by SDP regardless of the number of capacity expansion trajectories. Applying Random policy results in a non-zero Policy Difference that declines with an increase in the capacity expansion trajectories. Overall, using LP-relaxation or Mixed policies, *k*-ADP finds high-quality solutions while the Random policy results in less-favorable solutions.

In the analysis of results considering the Search Distance of *k*-ADP in Table 4, a larger δ value indicates that there are more neighboring states to the optimum result, hence, more options are available for the decision-maker. The *k*-ADP obtained an optimal solution similar to SDP in in-sample experiments when LP-relaxation and mix heuristic policies are applied, while the out-of-sample experimental outcomes are not influenced by the δ values. Applying a random heuristic policy in either experiment makes it unlikely to obtain good solutions because it selects the upper limit of the capacity expansion trajectory. Considering the out-of-sample demand path, larger δ values demonstrate a higher likelihood of finding the best decision. It is also observed that the Performance Gap and Policy Difference become smaller with an increase in δ values. Analyzing the performance of *R*-ADP in out-of-sample experiments, which is presented in Table 5, we observed that it finds the same average profit, 48938, with a standard deviation of 2.904 regardless of the number of demand and expansion trajectories.

Table 4. Results of the illustrative instance solved by *k*-ADP for various search distances.

Specification		Performance			
PEP	SD	AP	StD	PG (%)	PD
Random	1	48,739	2,904	0.408	1.00
Random	2	48,839	2,857	0.204	0.50
Random	3	48,839	2,857	0.204	0.50
Random	4	48,839	2,857	0.204	0.50
Random	5	48,839	2,857	0.204	0.50
LPR	1	48,938	2,904	0.000	0.00

LPR	2	48,938	2,904	0.000	0.00
LPR	3	48,938	2,904	0.000	0.00
LPR	4	48,938	2,904	0.000	0.00
LPR	5	48,938	2,904	0.000	0.00
Mix	1	48,938	2,904	0.000	0.00
Mix	2	48,938	2,904	0.000	0.00
Mix	3	48,938	2,904	0.000	0.00
Mix	4	48,938	2,904	0.000	0.00
Mix	5	48,938	2,904	0.000	0.00

PEP = Policy of Expansion Path; SD = Search Distance; AP = Average Profit; StD = Standard Deviation; PG = Performance Gap; PD = Policy Differences; Mix = Combination of Random and LP-relaxation.

Table 5. Result analysis of the illustrative instance considering various trajectories (*R*-ADP).

Factor	No.	NDT	NET	AP	StD	PG	PD
NDT changes	1	2	2	48,938	2,904	0.0000	0.00
	2	6	2	48,938	2,904	0.0000	0.00
	3	10	2	48,938	2,904	0.0000	0.00
	4	14	2	48,938	2,904	0.0000	0.00
	5	18	2	48,938	2,904	0.0000	0.00
NET changes	1	6	2	48,938	2,904	0.0000	0.00
	2	6	6	48,938	2,904	0.0000	0.00
	3	6	10	48,938	2,904	0.0000	0.00
	4	6	14	48,938	2,904	0.0000	0.00
	5	6	18	48,938	2,904	0.0000	0.00

NDT = Number of Demand Trajectory; NET = Number of Expansion Trajectory; AP = Average Profit; StD = Standard Deviation; PG = Performance Gap (%); PD = Policy Differences.

3.2.2. Real-world instance

The real-case example is a supply chain from the TFT-LCD industry with 2 Array manufacturers that accommodates 5 product families, A, B, C, D, and E. Specifications of the case study are summarized in Table 6. A planning period of 6 months is considered, and demand is divided into high- (H), moderate- (M), and low-demand (L). The probability of high demand remaining high is 0.6, while it may transition to moderate and low with probabilities of 0.3 and 0.1, respectively. The demand states and the transition probabilities are assumed to be similar for all product families within each planning period. Given two factories and five product families, there are $2\times5=10$ capacity variables in each state. Each factory can expand the production capacity up to 2 sets of masks for each product family with an exception for the first array manufacturer that can expand the production capacity of product family D to 3 sets. Capacity expansion cost for every product/array case is estimated at 400,000 USD per mask. Given four options for the capacity variable, i.e., an expansion for 1,2, or 3 units, there are $3^9\times4^1=78{,}732$ capacity states. Considering three demand states makes a total of $78732\times3=236{,}196$ combinations for each period and a total of $236196\times5=1{,}180{,}980$ states for the entire planning course.

Table 6. Specifications of the TFT-LCD case company.

Input	Factory	Product	1	2	3	4	5	6
Profit Margin (USD) in the past months	Array 1	14.1" (A)	18	18	18	18	18	18
		20.1" (B)	18	18	18	18	18	18
		22.0" (C)	20	20	20	20	20	20
		32.0" (D)	20	20	20	21	21	21
		37.0" (E)	18	18	18	19	19	19
	Array 2	14.1" (A)	18	18	18	18	18	18
		20.1" (B)	18	18	18	18	18	18
		22.0" (C)	20	20	20	20	20	20
		32.0" (D)	20	20	20	21	21	21
		37.0" (E)	18	18	18	19	19	19
Production Capacity (Sheet) in the past months	Array 1	14.1" (A)	16,000	16,000	16,000	16,000	16,000	16,000
		20.1" (B)	12,000	12,000	12,000	12,000	12,000	12,000
		22.0" (C)	12,000	12,000	12,000	12,000	12,000	12,000
		32.0" (D)	0	0	0	0	0	0
		37.0" (E)	12,000	12,000	12,000	12,000	12,000	12,000
	Array 2	14.1" (A)	18,000	18,000	18,000	18,000	18,000	18,000
		20.1" (B)	16,000	16,000	16,000	16,000	16,000	16,000
		22.0" (C)	16,000	16,000	16,000	16,000	16,000	16,000
		32.0" (D)	18,000	18,000	18,000	18,000	18,000	18,000
		37.0" (E)	16,000	16,000	16,000	16,000	16,000	16,000

Table 8 summarizes the results from the in-sample experiment. The SDP algorithm had to search from 1,180,980 feasible states with 7×10^6 function evaluations to find the best solution. The experiments show that *k*-ADP finds a solution very similar to that of SDP by searching only 0.05 percent of the state space, resulting in the average CPU Time of 51 seconds, which is 1.52 percent of the computational time required by SDP. Applying Rollout, *k*-ADP does not estimate the expected performance under various decisions, hence, the result is worse than applying Bellman. Table 7 also presents the out-of-sample experimental results. Regardless of applying Bellman or Rollout approaches, *k*-ADP obtains an average profit similar to that of SDP. The worst results from the Bellman approach are slightly better due to the applied repetitive searches.

Table 7. Performance analysis for in-sample and out-of-sample of the real-world instance.

Experiment	Method		Measure	OP	PG	ES	%	CPU
In-sample	SDP	-	-	119,885,311	-	1,180,980	-	3353
	k-ADP	Bellman	Ave.	120,139,859	4.71	555	0.05	44
			Max	128,915,871	7.27	598	0.05	47
			Min	107,228,869	10.56	501	0.04	37
		Rollout	Ave.	119,865,518	0.29	555	0.05	46
			Max	120,961,400	0.90	598	0.05	51
			Min	118,528,944	1.13	501	0.04	42
Out-of-sample	Method		Measure	AP	PG	StD		PD
	SDP	-	-	116,273,848	-	10,129,902		-
	k-ADP	Bellman	Ave.	116,223,198	0.052	10,121,604		1.516

		Max	116,273,848	0.000	10,129,902	0.000
		Min	116,166,662	0.110	10,093,544	2.000
	Rollout	Ave.	116,256,372	0.042	10,131,198	1.245
		Max	116,273,848	0.000	10,129,902	0.000
		Min	116,193,264	0.092	10,136,479	2.000
R-ADP	Bellman	Ave.	118,078,861	1.592	10,083,269	2.460
		Max	119,320,627	2.620	10,067,413	2.449
		Min	116,212,500	0.046	10,094,447	2.269
	Rollout	Ave.	116,211,860	0.053	10,092,854	2.370
		Max	116,221,340	0.045	10,093,183	2.279
		Min	116,201,474	0.062	10,093,178	2.436

OP = Optimum Profit; PG = Performance Gap in percentage; ES = Explored States; AP = Average Profit; PD = Policy Difference; OP = Optimum Profit; CPU = Computational time in seconds.

More details concerning the performance of *k*-ADP are provided in Table B2 of Appendix B. Assuming that the capacity expansion trajectory is a constant at 90 sets, the results approach those of the SDP when the demand path trajectory increases. In this situation, the mixed search policy results in the best outcomes, while applying LP-relaxation or Random search policy alone widens the gap between the obtained average profit and the best solution. Besides, an increase in the number of demand path trajectories results in longer computational times due to wider search scopes. It is worthwhile noting that the number of searched states and the computational time in the case of applying LP-relaxation is significantly shorter than mixed and random situations.

Considering a constant value for the number of demand trajectories while applying a mixed search policy, the quality of the obtained solution approaches that of the best solution obtained by SDP as the number of capacity expansion paths increases. When LP-relaxation is applied, the search procedure is not influenced by random sampling, hence, the obtained solution does not show significant differences as the number of capacity expansion trajectories increases. By applying the random policy, the chance factor has an impact on the search procedure and results in an unstable pattern. Concerning the number of search states and the computational time, applying LP-relaxation is more efficient than the Mixed and Random policies because an increase in the number of samples extends the search procedure.

Table 8 summarizes the optimal expected profit obtained under various search policies. The results under the Random policy are relatively worse compared to the other two approaches. Random search policy generates random solutions given the constraints, while LP-relaxation considers the demand trajectory extracted by simulations and applies the linear programming approach, hence, the solutions obtained are closer to those obtained by SDP. With an increase in the number of samples, the solution gap under LP-relaxation becomes smaller and approaches that of SDP. The Mixed policy results in better outcomes compared to the LP-relaxation approach because it combines the strength of LP-relaxation with the exploration capability of Random policy, which enables the algorithm to search for decisions that were ignored using the LP-relaxation approach.

Table 8. Result analysis of the real-world instance considering search distance (*k*-ADP).

Specification		Performance			
PEP	SD	AP	StD	PG (%)	PD

Random	1	115,606,344	10,135,741	0.574	5.45
Random	2	115,606,344	10,135,741	0.574	5.45
Random	3	115,606,344	10,135,741	0.574	5.45
Random	4	115,606,344	10,135,741	0.574	5.45
Random	5	115,606,344	10,135,741	0.574	5.45
LP-relaxation	1	116,166,662	10,136,479	0.092	2.00
LP-relaxation	2	116,166,662	10,136,479	0.092	2.00
LP-relaxation	3	116,166,662	10,136,479	0.092	2.00
LP-relaxation	4	116,166,662	10,136,479	0.092	2.00
LP-relaxation	5	116,166,662	10,136,479	0.092	2.00
Mix	1	116,005,884	10,508,970	0.230	3.26
Mix	2	116,079,474	10,514,495	0.167	3.00
Mix	3	116,124,555	10,515,534	0.128	2.82
Mix	4	116,158,999	10,471,488	0.099	2.77
Mix	5	116,158,999	10,471,488	0.099	2.01

PEP = Policy of Expansion Path; SD = Search Distance; AP = Average Profit; StD = Standard Deviation; PG = Performance Gap; PD = Policy Differences; Mix = Combination of Random and LP-relaxation.

Analyzing the out-of-sample experimental results demonstrates that an increase in the number of demand trajectories narrows down the gap between the average profit obtained by *k*-ADP and SDP when the capacity expansion trajectory is fixed at 90 sets and the Mixed and LP-relaxation search policies are applied. In the case of fixing the demand trajectory at 20, the results obtained by LP-relaxation and Mixed search policies are not influenced by the number of samples for the capacity expansion trajectory but improve the performance of the Random search policy. It is also observed that an increase in the number of demand trajectories or the number of the capacity expansion trajectory decreases the policy difference when a mixed search approach is applied. In this situation, it may be plausible to increase the net profit by purchasing more capacity expansion masks for different factories and products. From a search policy perspective, *k*-ADP's performance with the Mixed search approach is closer to that of SDP, and the Random approach results in the least stable outcomes due to the use of random numbers. The results considering changes in the Search Distance are reported in Table 9. It is observed that selecting a larger δ in *k*-ADP results in a greater number of neighbor states in the initial Markov Chain Monte Carlo Simulation. However, the difference between the neighboring states and the ignored state will be larger, and the decision may not be suitable for the ignored state. Overall, setting a suitable Search Distance is of high significance for an effective and efficient solution procedure.

This section considers 2 plant areas, 5 product families, and 6 phases during the planning period, and SDP can still find the best solution for capacity planning. Compare it with the results of the R-ADP proposed in Chapter 4 of this research. The purpose is to verify the model. It lies in the effectiveness and robustness of solving practical problems. The conclusions obtained from the experiment are as follows. Results obtained by *R*-ADP are provided in Table 9. Given a fixed number of samples for the capacity expansion trajectory, the average net profit value obtained by *R*-ADP gradually approaches that of SDP with an increase in the number of samples for

demand trajectory and shows a state of convergence when it reaches 50. Under the same condition, Policy Difference gradually decreases. It is also evident that an increase in the number of samples for demand trajectory under fixed expansion trajectory increases the computational time. Considering changes in the number of samples for capacity expansion trajectories when the demand path is fixed at 50, the average net profit value obtained by *R*-ADP gradually approaches that of SDP and the Policy Difference decreases. Expectedly, an increase in the number of samples for capacity expansion trajectories imposes a longer computational time.

Table 9. Result analysis of real-world instance considering various trajectories (*R*-ADP).

Factor	No.	NDT	NET	AP	StD	PG	PD	CPU
NDT changes	1	10	90	116,218,572	10,082,420	0.048%	2.450	22206
	2	20	90	116,220,662	10,100,319	0.046%	2.298	28920
	3	30	90	116,225,517	10,097,355	0.042%	2.294	39720
	4	40	90	116,225,694	10,094,124	0.041%	2.224	50460
	5	50	90	116,227,040	10,096,895	0.040%	2.211	60900
NET changes	1	50	10	116,186,233	10,079,647	0.075%	2.402	33600
	2	50	30	116,196,619	10,090,116	0.066%	2.401	38820
	3	50	50	116,201,533	10,095,475	0.062%	2.365	45531
	4	50	70	116,216,478	10,106,088	0.049%	2.255	56742
	5	50	90	116,227,040	10,096,895	0.040%	2.211	60900

NDT = Number of Demand Trajectory; NET = Number of Expansion Trajectory; AP = Average Profit; StD = Standard Deviation; PG = Performance Gap (%); PD = Policy Differences; CPU = Computational time (Sec).

When comparing the performance of the developed ADP methods, *R*-ADP outperforms *k*-ADP for the number of samples under a certain value regardless of changes in the demand or capacity expansion trajectories. When the number of samples goes beyond a certain value, *k*-ADP performs significantly better than *R*-ADP. Overall, performance improvement in *k*-ADP with changes in parameters is more apparent, that is, *R*-ADP is less sensitive to parameter changes when compared to *k*-ADP.

3.2.3. Large instances

Results analysis of the large instances is provided in Table 12. Investigating the first large instance with 2 factories, 5 product families, 6 planning periods, and independent demand status, it took about 2.5 days for SDP to find the best solution, while the *R*-ADP and *k*-ADP needed 1.31 and 0.29 days, respectively, to obtain a very close approximation to the optimal capacity expansion solution. It is observed that *k*-ADP performs better in terms of Performance Gap and Policy Difference when compared to *R*-ADP. The experiments also revealed that SDP is not capable of solving instances with six and more than six products in a reasonable time.

Table 10. Numerical results of the solved large-scale instances.

Instance	Method	AP	PG (%)	StD	PD	CPU
A	SDP	123,353,000	-	95,659,380	-	262,656
	k-ADP	123,185,120	0.015	4,477,929	1.821	25,760
	R-ADP	122,971,353	0.189	4,431,834	2.876	114,008

B	SDP	N/A	-	N/A	-	>1 week
	k-ADP	155,733,795	-	19,033,661	-	5,036
	R-ADP	155,224,449	-	18,817,809	-	116,812
C	SDP	N/A	-	N/A	-	>1 week
	k-ADP	177,121,930	-	20,823,590	-	13,896
	R-ADP	169,524,771	-	20,823,590	-	201,600

AP = Average Profit; PG = Performance Gap; StD = Standard Deviation; PD: Performance Difference; CPU = Computational time in seconds.

The computational complexity of the test instances and the performance of the algorithms in terms of computational time are analyzed in Table 11. The analysis shows that SDP requires extensively more computational time compared to *k*-ADP and *R*-ADP in all the test instances. It is observed that growth in the number of states and decision variables makes it prohibitive to find an optimal solution using SDP. Taking the case comprising 2 plants, 5 product families, and 6 planning periods with heterogeneous demands as an example, SDP required approximately 2.5 days to find the optimum solution in a full-enumeration method. Searching only 0.031 percent of the entire solution space, the *k*-ADP method required less than 0.1 percent of the time required by SDP to find the best solution. In larger instances, SDP is no longer capable of obtaining the optimum result in a reasonable time. Given the situation with 3 production sites, 6 product families, and 6 planning periods with homogeneous demands, where a total of 7.75×10^{9} states were identified, it takes about 1.4 hours for *k*-ADP to obtain a set of near-optimal capacity expansion solutions. Finally, considering the problem of 3 plants, 7 product families, 6 planning periods, and homogeneous product demands, which results in a total of 2.09×10^{11} states, *k*-ADP yielded the near-optimum solution in 3.86 hours and SDP failed to finish computations in a reasonable time. For solving the largest instance comprising 3 factories, 7 product families, and 6 planning periods, *R*-ADP required 2.33 days of computation time while *k*-ADP yields a slightly better solution in only 0.16 days. Overall and considering *k*-ADP and *R*-ADP's performance, one can conclude that the developed dynamic programming approaches are superior to SDP.

Table 11. Computational performance analysis of the algorithms (best in **bold**).

Instance	Configuration (Site, product, period, demand)	State Space	Action Space	Computational complexity	Computational time (Sec)		
					SDP	*k*-ADP	*R*-ADP
1	2,2,4, Homogeneous	96	16	486	3	**1**	30
2	2,5,6, Homogeneous	1.18×10^{6}	7.87×10^{3}	7.00×10^{6}	3,353	**46**	28,920
3	2,5,6, Heterogeneous	9.57×10^{7}	7.87×10^{6}	5.67×10^{8}	262,656	**25,760**	114,008
4	3,6,6, Homogeneous	7.75×10^{9}	5.17×10^{8}	1.18×10^{13}	>1 week	**5,036**	116,812
5	3,7,6, Homogeneous	2.09×10^{11}	1.39×10^{10}	2.54×10^{15}	>1 week	**13,896**	201,600

3.3. Statistical results

The developed approximation algorithms, *k*-ADP and *R*-ADP, demonstrated to be significantly more efficient than SDP. It is also observed that *k*-ADP requires a shorter computational time compared to *R*-ADP. Statistical analysis is conducted to investigate whether the solutions' quality of *k*-ADP and *R*-ADP are significantly different considering various situations. For this purpose, a statistical test of significance is applied considering various dimensions, i.e., demand changes,

initial production capacity, and expansion cost. Statistical results are illustrated in Table 14. In situations characterized by low and high-profit margins, there is a significant difference between the performance of the algorithms with *k*-ADP obtaining better results compared to *R*-ADP. Considering the initial capacity as the gauge in Table 12, it is confirmed that *k*-ADP outperforms *R*-ADP. The same claim is true when low and high-capacity expansion costs are considered. The resulting *p*-values confirm the above assertions.

Table 12. Computational performance analysis of the algorithms (best in **bold**).

Consideration	Instance	Paired difference (*k*-ADP Vs. *R*-ADP)				*T*	*p-value*
		N	Mean	StD	SEM		
Profit Margin	Low	1200	121667	325909	9408	12.93	0.000
	High	1200	31149.9	162451.5	4689.6	6.64	0.000
Initial Capacity	Low	1200	70626.1	229091.2	6613.3	10.68	0.000
	High	1200	82190.8	290098.9	8374.4	9.81	0.000
Expansion Cost	Low	1200	17281.7	49592.5	1431.6	12.07	0.000
	High	1200	135535	356721	10298	13.16	0.000
Demand Transition	Stable	1200	-6829	71640.4	2532.87	-2.7	0.007
	Negative	1200	132848	394079	13933	9.53	0.000
	Positive	1200	-167.9	94825.6	3352.6	-0.05	0.960

Paired *t*-test results in the Table also confirm that there is no significant difference between the performance of *k*-ADP and *R*-ADP when dealing with industry situations with positive demand transitions. This result highlights the role of randomness in planning unstable market situations. Given that the results obtained by *k*-ADP are generally better than those of *R*-ADP, one can conclude that *k*-ADP is more effective and computationally more efficient. *k*-ADP can be considered as a strong benchmark for further developments and dynamic programming applications in other strategical, tactical, and operational decisions in the supply chain.

Our findings have several implications for the manufacturing supply chains. With the customers expecting more diverse products that are cheaper, delivered on time, and are often ordered more frequently and in smaller quantities, companies are increasingly under pressure to adjust the product mix and pursue just-in-time production. In this situation, market volatilities make capacity adjustment decisions inevitable. Production managers need flexible capacity planning platforms for timely adjustment in the number of workforce and machinery. The major value of the proposed approach is in its ability to solve large-scale problems much more efficiently compared to the state-of-the-art. Besides, the robustness of the approach improves the reliability of the outcomes for well-informed capacity management decisions.

Production capacity planning for a system that is horizontally integrated and has a rather high demand mix helps reduce the chances of sub-optimal decisions. As a prime example of having horizontally integrated systems, while a larger capacity for one machine or production plant may result in better local outcomes, it may not necessarily result in a global best. Having that said, the timely adjustment of the production capacity across distributed manufacturing systems not only reduce the odds of lost orders and backlogs, but also improves cost-efficiency,

particularly for inventory management and inbound logistics. On the other hand, it can reduce the workload for better production scheduling outcomes and shorter cycle times, while ensuring high utilization of the resources. Besides, with Industry 5.0 emphasizing more sustainable and human-centric production solutions, a robust production capacity planning platform will have implications for reducing the overtime work that is inevitable in overly tight production capacities. In this situation, a combination of flexible production capacity with new technologies may be crucial for manufacturers that face high demand volatilities. Concerning demand mix volatilities, taking into account the similarities between products may further improve the outcomes of production capacity planning decisions. For this purpose, learning-based techniques can help improve the realization of these similarities and facilitate well-informed capacity allocation decisions.

4. Concluding remarks

As a major supply chain decision, production capacity planning and allocation establish the tactical frame for the company's mid- to long-term operations. Taking into account the uncertainties and volatility of markets, capacity adjustment as an operational strategic decision enables the company to stay competitive regardless of the sort of competitive strategy they pursue. This study explored the capacity planning decisions by developing two computationally efficient ADP methods. Using real data from the TFT-LCD industry and several large-scale instances, the efficiency of the proposed approaches for saving computational resources and real-time decision analysis were examined.

Considering 100 sets of out-of-sample demand scenarios each of which featuring a complexity of 5.67×10^{8} function evaluations, *k*-ADP and *R*-ADP algorithms explored only 0.031% of the solution space to find the (near-) optimal outcome requiring 9.81 and 43.41 percent of the computational time required by the benchmark algorithm, respectively. We observed that with an increase in the search distance up to a certain extent, *k*-ADP's result approaches the true average net profit value obtained by the full-enumeration approach. We also showed that the developed methods can effectively solve the instances that cannot be solved using SDP. Overall, the numerical results indicated that *k*-ADP and *R*-ADP are significantly more effective than SDP. Besides, *R*-ADP is less sensitive to parameter changes, which makes it a more robust approach.

This study can be extended in several directions. The present study considered the price of each product family as a fixed value, while in practice product margins may vary according to demand, which may result in overly optimistic or pessimistic capacity planning decisions. As the first possible direction for future research, one can consider price uncertainty to explore the influence of the demand-price relationships on production capacity decisions. From the Operations Research viewpoint, the model's performance value can account for risk-related measures. As a second suggestion for future research, the minimum profit that can be obtained from the capacity planning decision to measure different external factors considering a tolerable confidence level. In this situation, the simulation results under the mutated environment improve the robustness of the model. Finally, the Concept of Stratification and Incremental Enlargement can be tested and compared to the ADP approaches developed in this study.

Compliance with ethical standards

Conflict of interest

The authors declare that they have no known conflict of interest.

References

Alasaly M, Mathkour H, Al-Azzoni I (2015) Dynamic Capacity Planning with Comprehensive Formation of SLAs in Clouds. Int J Comput Appl 117:

Atherton RW (1989) Dynamic Capacity Planning Using Simulation Models. In: Proceedings of the Fourth Symposium on Automated Integrated Circuit Manufacturing

Bunn DW, Oliveira FS (2016) Dynamic capacity planning using strategic slack valuation. Eur J Oper Res 253:40–50

Ceryan O, Koren Y (2009) Manufacturing capacity planning strategies. CIRP Ann 58:403–406

Chen Y-Y, Chen T-L, Liou C-D (2013) Medium-term multi-plant capacity planning problems considering auxiliary tools for the semiconductor foundry. Int J Adv Manuf Technol 64:1213–1230

Choi J, Realff MJ, Lee JH (2004) Dynamic programming in a heuristically confined state space: a stochastic resource-constrained project scheduling application. Comput Chem Eng 28:1039–1058

Choi J, Realff MJ, Lee JH (2006) Approximate dynamic programming: application to process supply chain management. AIChE J 52:2473–2485

Chopra S, Meindl P (2015) Supply Chain Management Strategy and Operation. Pearson 13–17

Deif A, ElMaraghy HA (2009) Dynamic capacity planning and modeling its complexity. In: Changeable and Reconfigurable Manufacturing Systems. Springer, pp 227–246

Dubey SK (2020) Systems, computer-readable media and computer-implemented methods for automated, dynamic capacity planning using http response header fields

Georgiadis P (2013) An integrated System Dynamics model for strategic capacity planning in closed-loop recycling networks: A dynamic analysis for the paper industry. Simul Model Pract Theory 32:116–137

Hu Y, Liu Y, Wang Z, et al (2020) A two-stage dynamic capacity planning approach for agricultural machinery maintenance service with demand uncertainty. Biosyst Eng 190:201–217

Katanyukul T, Duff WS, Chong EKP (2011) Approximate dynamic programming for an inventory problem: Empirical comparison. Comput Ind Eng 60:719–743

Kingsman BG (1997) Input/output backlog control and dynamic capacity planning in versatile manufacturing companies. In: Stochastic Modelling in Innovative Manufacturing. Springer, pp 97–122

Kingsman BG (2000) Modelling input--output workload control for dynamic capacity planning in production planning systems. Int J Prod Econ 68:73–93

La Torre D, Abdelaziz F Ben, Alaya H (2019) Dynamic programming and optimal control for vector-valued functions: A State-of-the-art review. RAIRO-Operations Res

Li C, Liu F, Cao H, Wang Q (2009) A stochastic dynamic programming based model for uncertain production planning of re-manufacturing system. Int J Prod Res 47:3657–3668

Lin JT, Chen T-L, Chu H-C (2014) A stochastic dynamic programming approach for multi-site capacity planning in TFT-LCD manufacturing under demand uncertainty. Int J Prod Econ 148:21–36. https://doi.org/10.1016/j.ijpe.2013.11.003

Lin JT, Wu C-H, Chen T-L, Shih S-H (2011) A stochastic programming model for strategic capacity planning in thin film transistor-liquid crystal display (TFT-LCD) industry. Comput Oper Res 38:992–1007

Martinez-Costa C, Mas-Machuca M, Benedito E, Corominas A (2014) A review of mathematical programming models for strategic capacity planning in manufacturing. Int J Prod Econ 153:66–85

Pehlivan C, Augusto V, Xie X (2014) Dynamic capacity planning and location of hierarchical service networks under service level constraints. IEEE Trans Autom Sci Eng 11:863–880

Pourhejazy P, Kwon OK, Chang Y-T, Park H (2017) Evaluating resiliency of supply chain network: A data envelopment analysis approach. Sustain 9:. https://doi.org/10.3390/su9020255

Pourhejazy P, Sarkis J, Zhu Q (2020) Product deletion as an operational strategic decision: Exploring the sequential effect of prominent criteria on decision-making. Comput Ind Eng 140:106274. https://doi.org/10.1016/j.cie.2020.106274

Pratikakis NE, Lee JH, Realff MJ (2006) A real time adaptive dynamic programming approach for planning and scheduling. In: Computer Aided Chemical Engineering. Elsevier, pp 1179–1184

Pratikakis NE, Realff MJ, Lee JH (2010) Strategic capacity decision-making in a stochastic manufacturing environment using real-time approximate dynamic programming. Nav Res Logist 57:211–224

Seidel G, Preuss P, Canbolat C, et al (2019) An integration of static and dynamic capacity planning for a ramping fab. In: 2019 Winter Simulation Conference (WSC). pp 2304–2311

Sureka S (2019) Dynamic capacity planning for application delivery platform across multiple cloud deployment

Tarek A, Elsayed H, Rashad M, et al (2020) Dynamic Programming Applications: A Suvrvey. In: 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES). pp 380–385

Vlachos D, Georgiadis P, Iakovou E (2007) A system dynamics model for dynamic capacity planning of remanufacturing in closed-loop supply chains. Comput Oper Res 34:367–394

Voelkel MA, Sachs A-L, Thonemann UW (2020) An aggregation-based approximate dynamic programming approach for the periodic review model with random yield. Eur J Oper Res 281:286–298

Wu C-H, Chuang Y-T (2010) An innovative approach for strategic capacity portfolio planning under uncertainties. Eur J Oper Res 207:1002–1013

Wu SD, Erkoc M, Karabuk S (2005) Managing capacity in the high-tech industry: A review of literature. Eng Econ 50:125–158

Appendices

Appendix A

```
integer t;                  // Period, t={0,1,...,T}
integer k;                  // Product, k={1,2,...,K}
integer de_s[k][t];     // Demand Scenario of product k in period t
double p[k][i][j];      // Demand state transition probability of product k   i,j={L,M,H}
double r1;                  // Random Number of Demand Scenario

Input:

     p[k][i][j]        for all k={1,2,...,K) ,  i,j={L,M,H) ;          // Demand state transition probability

Initialize:

         for (k=1 to K ){
              de_s[k][0]='H';       // Initial demand state
         }

Generate_Demand_Sample_Path:

for (t=1 to T) {

     for ( k=1 ot K ){

          r1 = Uniform(0,1)             //Generate random number

          if ( 0 < r1 <= ( p[de_s[k][t-1]][L] ))
               de_s[k][t] = 'L' ;
          elseif ( ( p[de_s[k][t-1]][L] ) < r1 <= ( p[de_s[k][t-1]][L] + p[de_s[k][t-1]][M] ) )
               de_s[k][t] = 'M' ;
          elseif ( ( p[de_s[k][t-1]][L] + p[de_s[k][t-1]][M] ) < r1 <=  1 )
               de_s[k][t] = 'H' ;

     }
}


Output:

        de_s[t]  for all
```

Figure A1. Pseudocode of the Markov Chain Monte Carlo Simulation module.

Appendix B

Table B1. Results analysis of the illustrative instance solved by *k*-ADP for various trajectories.

Factor	In-sample									Out-of-sample							
	State				Performance					State				Performance			
	No.	NDT	NET	PEP	EP	PG	ES	ES %	CPU	No.	NDT	NET	PEP	AP	StD	PG	PD
NDT changes	1	2	10	RND	46335	6.522	28	29.167	1	1	2	10	RND	48,490	2,904	0.917	1.41
	2	6	10	RND	50925	2.738	26	27.083	1	2	6	10	RND	48,558	2,911	0.778	2.73
	3	10	10	RND	49225	0.692	33	34.375	1	3	10	10	RND	48,651	2,904	0.587	2.00
	4	14	10	RND	49424	0.291	27	28.125	1	4	14	10	RND	48,666	2,904	0.557	1.00
	5	18	10	RND	49457	0.224	37	38.542	1	5	18	10	RND	48,601	2,915	0.690	1.73
	6	2	10	LPR	49291	0.559	5	5.208	1	6	2	10	LPR	48,938	2,904	0.000	0.00
	7	6	10	LPR	49568	0.000	13	13.542	1	7	6	10	LPR	48,938	2,904	0.000	0.00
	8	10	10	LPR	49568	0.000	13	13.542	1	8	10	10	LPR	48,938	2,904	0.000	0.00
	9	14	10	LPR	49568	0.000	13	13.542	1	9	14	10	LPR	48,938	2,904	0.000	0.00
	10	18	10	LPR	49568	0.000	13	13.542	1	10	18	10	LPR	48,938	2,904	0.000	0.00
	11	2	10	Mixed	49568	0.000	41	42.708	1	11	2	10	Mixed	48,938	2,904	0.000	0.00
	12	6	10	Mixed	49568	0.000	43	44.792	1	12	6	10	Mixed	48,938	2,904	0.000	0.00
	13	10	10	Mixed	49568	0.000	45	38.542	1	13	10	10	Mixed	48,938	2,904	0.000	0.00
	14	14	10	Mixed	49568	0.000	49	51.042	1	14	14	10	Mixed	48,938	2,904	0.000	0.00
	15	18	10	Mixed	49568	0.000	55	57.292	1	15	18	10	Mixed	48,938	2,904	0.000	0.00
NET changes	1	14	2	RND	49189	0.765	11	11.458	1	1	14	2	RND	48,558	2,911	0.778	2.73
	2	14	6	RND	50881	2.649	19	19.792	1	2	14	6	RND	48,720	2,911	0.447	2.83
	3	14	10	RND	50925	2.738	26	27.083	1	3	14	10	RND	48,763	2,915	0.360	2.41
	4	14	14	RND	50916	2.719	32	33.333	1	4	14	14	RND	48,761	2,904	0.363	1.41
	5	14	18	RND	49350	0.440	53	55.208	1	5	14	18	RND	48,877	2,921	0.125	1.41
	6	14	2	LPR	49568	0.000	13	13.542	1	6	14	2	LPR	48,938	2,904	0.000	0.00
	7	14	6	LPR	49568	0.000	13	13.542	1	7	14	6	LPR	48,938	2,904	0.000	0.00
	8	14	10	LPR	49568	0.000	13	13.542	1	8	14	10	LPR	48,938	2,904	0.000	0.00
	9	14	14	LPR	49568	0.000	13	13.542	1	9	14	14	LPR	48,938	2,904	0.000	0.00
	10	14	18	LPR	49568	0.000	13	13.542	1	10	14	18	LPR	48,938	2,904	0.000	0.00
	11	14	2	Mixed	49568	0.000	25	26.042	1	11	14	2	Mixed	48,938	2,904	0.000	0.00
	12	14	6	Mixed	49568	0.000	35	36.458	1	12	14	6	Mixed	48,938	2,904	0.000	0.00
	13	14	10	Mixed	49568	0.000	43	44.792	1	13	14	10	Mixed	48,938	2,904	0.000	0.00
	14	14	14	Mixed	49568	0.000	57	59.375	1	14	14	14	Mixed	48,938	2,904	0.000	0.00
	15	14	18	Mixed	49568	0.000	55	57.292	1	15	14	18	Mixed	48,938	2,904	0.000	0.00

NDT = Number of Demand Trajectory; NET = Number of Expansion Trajectory; PEP= Policy of Expansion Path; EP = Expected Profit; PG = Performance Gap (%); ES = Explored States; CPU = Computational time (Sec); AP = Average Profit; StD = Standard Deviation; PD = Policy Differences; LPR = LP-relaxation; RND = Random; Mixed = combination of Random and LP-relaxation.

Table B2. Result analysis of the real-world instance considering various trajectories (*k*-ADP).

Factor	In-sample									Out-of-sample							
	State				Performance					State				Performance			
	No.	NDT	NET	PEP	EP	PG	ES	% ES	CPU	No.	NDT	NET	PEP	AP	StD	PG	PD
NDT changes	1	10	90	RND	121,541,282	1.381	678	0.057	64	1	10	90	RND	115,704,885	10,097,238	0.489	5.29
	2	20	90	RND	118,812,298	0.895	766	0.065	70	2	20	90	RND	115,549,329	10,135,741	0.623	5.00
	3	30	90	RND	119,089,130	0.664	760	0.064	75	3	30	90	RND	115,472,451	10,135,741	0.689	5.02
	4	40	90	RND	119,253,354	0.527	754	0.064	79	4	40	90	RND	115,636,131	10,137,162	0.549	5.97
	5	50	90	RND	119,320,627	0.471	808	0.068	83	5	50	90	RND	115,703,636	10,136,479	0.490	6.24
	6	10	90	LPR	123,105,886	2.686	55	0.005	8	6	10	90	LPR	116,166,662	10,136,479	0.092	2.00
	7	20	90	LPR	120,505,024	0.517	73	0.006	13	7	20	90	LPR	116,248,346	10,126,087	0.022	1.09
	8	30	90	LPR	120,338,437	0.378	100	0.008	15	8	30	90	LPR	116,273,848	10,129,902	0.000	0.00
	9	40	90	LPR	119,195,537	0.575	121	0.010	22	9	40	90	LPR	116,273,848	10,129,902	0.000	0.00
	10	50	90	LPR	119,364,020	0.435	124	0.010	25	10	50	90	LPR	116,273,848	10,129,902	0.000	0.00
	11	10	90	Mix	120,550,965	0.555	646	0.055	65	11	10	90	Mix	116,168,258	10,455,603	0.091	3.02
	12	20	90	Mix	119,825,254	0.050	805	0.068	79	12	20	90	Mix	116,259,909	10,114,886	0.012	1.52
	13	30	90	Mix	119,918,023	0.027	883	0.075	90	13	30	90	Mix	116,273,848	10,129,902	0.000	0.00
	14	40	90	Mix	119,778,653	0.089	880	0.075	94	14	40	90	Mix	116,273,848	10,129,902	0.000	0.00
	15	50	90	Mix	119,861,817	0.020	901	0.076	108	15	50	90	Mix	116,273,848	10,129,902	0.000	0.00
NET changes	1	50	10	RND	117,521,500	1.972	124	0.010	18	1	50	10	RND	115,319,069	10,137,162	0.821	5.74
	2	50	30	RND	120,589,522	0.587	304	0.026	31	2	50	30	RND	115,526,955	10,137,162	0.642	7.89
	3	50	50	RND	120,108,281	0.186	469	0.040	45	3	50	50	RND	115,544,755	10,120,894	0.627	8.24
	4	50	70	RND	119,971,671	0.072	586	0.050	59	4	50	70	RND	115,578,235	10,135,741	0.598	6.38
	5	50	90	RND	119,320,627	0.471	808	0.068	83	5	50	90	RND	115,549,329	10,135,741	0.623	5.00
	6	50	10	LPR	120,338,437	0.378	124	0.010	25	6	50	10	LPR	116,243,079	10,129,902	0.027	1.00
	7	50	30	LPR	120,338,437	0.378	124	0.010	25	7	50	30	LPR	116,248,346	10,126,087	0.022	1.09
	8	50	50	LPR	120,338,437	0.378	124	0.010	25	8	50	50	LPR	116,248,346	10,126,087	0.022	1.09
	9	50	70	LPR	120,338,437	0.378	124	0.010	25	9	50	70	LPR	116,248,346	10,126,087	0.022	1.09
	10	50	90	LPR	120,338,437	0.378	124	0.010	25	10	50	90	LPR	116,248,346	10,126,087	0.022	1.09
	11	50	10	Mix	121,155,711	1.060	211	0.018	30	11	50	10	Mix	116,159,762	10,454,530	0.098	3.05
	12	50	30	Mix	120,012,121	0.106	436	0.037	41	12	50	30	Mix	116,166,662	10,136,479	0.092	2.00
	13	50	50	Mix	119,699,212	0.155	583	0.049	52	13	50	50	Mix	116,250,348	10,091,756	0.020	1.46
	14	50	70	Mix	119,949,134	0.053	787	0.067	82	14	50	70	Mix	116,243,079	10,129,902	0.027	1.00
	15	50	90	Mix	119,861,817	0.020	901	0.076	108	15	50	90	Mix	116,259,909	10,114,886	0.012	1.52

NDT = Number of Demand Trajectory; NET = Number of Expansion Trajectory; PEP= Policy of Expansion Path; EP = Expected Profit; PG = Performance Gap (%); ES = Explored States; CPU = Computational time (Sec); AP = Average Profit; StD = Standard Deviation; PD = Policy Differences; LPR = LP-relaxation; RND = Random; Mix = Combination of Random and LP-relaxation.