Computers & Industrial Engineering 196 (2024) 110404

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

An online dynamic dual bin packing with lookahead approach for server-to-cell assignment in computer server industry

Mahmud Parvez [a], Pratik J. Parikh [a,*], Faisal Aqlan [a], Md. Noor-E-Alam [b]

[a] Department of Industrial Engineering, University of Louisville, Louisville, KY, United States
[b] Department of Mechanical and Industrial Engineering, Northeastern University, Boston, MA, United States

ARTICLE INFO

Keywords:
Dynamic assignment
Dual bin packing
Lookahead
Optimization
Genetic algorithm
Computer server industry

ABSTRACT

Efficient production scheduling in computer server industry remains a critical challenge. This challenge becomes particularly acute during the testing phase, where servers are assigned for testing in a modular environment consisting of test banks. Unique characteristics such as online arrival of servers known only over a short time window, skewed arrival pattern, power and cooling compatibility constraints, and dynamic assignment complicate decision making, a situation we observed at our industry partner's site, which often leads to missing due dates and loss of customer trust. Motivated by this challenge, we introduce an Online Dynamic Dual Bin Packing with Lookahead problem and propose an integer linear programming model that maximizes the number of assigned servers. To efficiently solve this model, we decompose the problem into a set of subproblems for a given lookahead length. A '2-phase' computational framework is proposed that seamlessly integrates mathematical programming with genetic algorithm. Based on realistic data available from a server manufacturer, our findings suggest that solutions are sensitive to (i) the length of the lookahead window, (ii) testing capacity and server arrival pattern, (iii) test processing time requirement, and (iv) physical allocation of servers.

1. Introduction

The emergence of cutting-edge electronic devices with shorter lifespans, combined with advancements in information technology (IT), has spurred the rise in demand for electronic products. The U.S. electronics manufacturing landscape is diverse and encompasses numerous industry leaders such as IBM, Asus, Dell, HP, Huawei, LG, Microsoft, Ricoh, and Xerox. The environment is characterized by aggressive introduction cycles of new products, extreme demand skewness, and significant engineering changes (Aqlan, Lam, & Ramakrishnan, 2014; Saha, 2015). These businesses, however, face challenges such as supply–demand imbalance, insufficient quality management practices, large inventories, poor spare parts information, and insufficient automation (Kurilova-Palisaitiene, Sundin, & Poksinska, 2018; Nasr, Haselkorn, Parnel, Burn, & Hanson, 2017).

1.1. Motivation

This work is motivated by our interaction with a leading computer server manufacturer in the US. The company uses a Configure-To-Order (CTO) environment, as illustrated in Fig. 1. The CTO production setup is a hybrid approach combining elements from both build-to-plan and assemble-to-order systems, often referred to as the fabrication-fulfillment strategy. During the fabrication phase, components or subassemblies undergo production, testing, and assembly based on forecasted production plans. These components remain in storage until a specific customer order arrives. In the fulfillment phase, assembly of the final products is carried out based on real-time customer demands.

In such an environment, the assembly of servers involves components that are not only expensive, but also requires comprehensive testing processes to meet essential quality and reliability standards before dispatch. The production lines in this setting rely on multi-tiered suppliers, both internal and external, often characterized by extended lead times for supplies (Aqlan et al., 2014).

1.2. Online decision-making for fulfillment test

The fulfillment test is the key final step for testing the assembled servers to ensure the highest quality before shipping. During this step, computer servers undergo rigorous testing within a modular test environment that consists of test banks, each comprising numerous test cells, to ensure their quality, performance, and reliability. The set of the

* Corresponding author at: Department of Industrial Engineering, 132 Eastern Parkway, University of Louisville, Louisville, KY 40292, United States.
E-mail address: pratik.parikh@louisville.edu (P.J. Parikh).

https://doi.org/10.1016/j.cie.2024.110404
Received 3 April 2024; Received in revised form 16 July 2024; Accepted 17 July 2024
Available online 20 July 2024

testing facility at our partners site includes four test banks, each with fourteen dedicated test cells for testing the servers (Fig. 2). Each test bank has two cooling units, and test cells have specific voltage and cooling capabilities.

In a CTO setting, this testing phase is characterized by significant skewness in server (customer order) arrival with certainty over a short time window, strict power and cooling compatibility requirements, and the availability of test cells (resources) after a server departs upon test completion. These factors significantly impact the overall manufacturing workflow, leading to a higher number of unfinished servers. These unique characteristics of the fulfillment test underscore the need for precise and careful scheduling of servers to the test cells, a critical step in streamlining the manufacturing process and meeting the increasing demand for servers.

At our partner organization's facility, the decision-making process is manual and online as information about server arrival is only available with certainty over a short time-window during the planning horizon. Whenever a new server arrives, the decision makers at this facility immediately allocate it to a current (or soon-to-be-available) test cell. Such a myopic approach makes it challenging for ensuring that no servers are unassigned or partially finished by the server due date, and the test cells are effectively utilized. Unassigned or unfinished servers can lead to missing due dates and customer trust. While missing customer orders is not allowed at our partner's facility, even if it were to be outsourced to a third-party with enough capacity, then it would result in delays or financial loss.

Addressing the shortcomings of this current myopic decision-making approach, characterized by the first-come-first-serve assignment of servers to cells, we recognize that such practices may not be necessary if information about server arrivals was known with certainty, even over a few future periods. We define this foresight as 'lookahead' and explore it in tandem with the online dynamic assignment of servers to cells.

1.3. Research questions and contributions

Realizing that this problem of online decision making is not unique to just this server manufacturing company, but to many others with a similar testing setup, we attempt to address the following generic research questions:

- How to optimally assign servers to test cells in a given lookahead window to minimize the number of unfinished servers?
- How does the choice of lookahead window affect solution quality?
- How do other system parameters such as server arrival pattern, server size, and system capacity affect solution quality?

Through these questions, we will add the following contributions to scientific literature in this area. First, we propose an Online Dynamic Dual Bin Packing with Lookahead (OD-DBP-LA) problem, which generalizes the Online Bin Packing with Lookahead (OBP-LA) problem proposed by Dunke et al. (2016). The generalization is along three directions:

- the time lookahead (LA) concept, where the lookahead is determined by the length of a time-window during which the server arrival is known with certainty;
- the dual bin packing (DBP) concept, where the aim is to maximize the number of items that can be packed in available bins (Assmann, Johnson, Kleitman, & Leung, 1984; Labbé et al., 1995; Peeters & Degraeve, 2006; Parikh et al., 2008; Vijayakumar, Parikh, Scott, Barnes, & Gallimore, 2013); in our problem, servers are treated as items, test cells as bins, and our aim is to maximize the number of assigned servers; and
- the dynamic concept (D), where servers depart the test cells after they are processed, which frees up the cells for another assignment; i.e., this is not a static assignment (Coffman, Garey, & Johnson, 1983).

Second, we propose an integer linear programming model for the OD-DBP-LA that (i) prioritizes servers with earlier due dates, (ii) incorporates system-related and demand-related constraints, and (iii) accounts for sequential and dynamic aspects of the decision-making problem capturing the interdependencies and the cascading effects of decisions throughout the time-horizon. The objective is to minimize the number of unfinished servers, a key requirement in server manufacturing industry. Third, given the challenges of solving industry-scaled problems with large problem sizes and longer lookahead windows, traditional solvers fail to provide optimal solutions within a reasonable timeframe. This is a significant issue since decision-makers have limited time for model execution and decision-making. To address this, we propose a '2-phase' computational framework for efficiently solving each deterministic, lookahead window-specific subproblems. The proposed '2-phase' computational framework seamlessly integrates a metaheuristic based on genetic algorithm (GA) framework with mathematical programming approach.

Our comprehensive numerical analysis shows the efficacy of the proposed '2-phase' computational framework, particularly in solving large-scale instances with longer lookahead windows. Furthermore, the analysis reveals a notable sensitivity of solution quality to the length of the lookahead window, with longer windows consistently yield fewer unfinished servers compared to shorter ones. Moreover, the variability in test processing times and the proportion of servers requiring two test cells (2TC) significantly influence the characteristics of unfinished servers, especially as the lookahead window shortens. Alongside unfinished servers, we also calculated, post-hoc, a secondary measure; the number of tardy servers. The server arrival pattern exhibits a significant impact on the performance metrics: right-skewed arrival patterns tend to result in fewer unfinished but more tardy servers, while left-skewed arrivals lead to more unfinished and fewer tardy servers. Additionally,

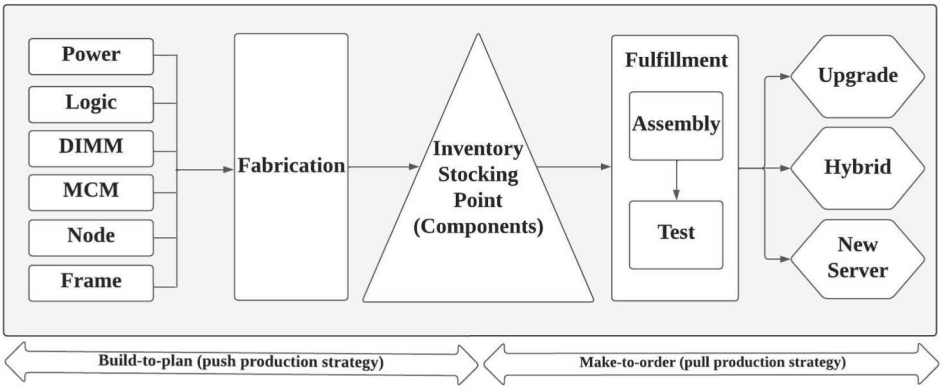


Fig. 1. Overview of the server manufacturing system: Fabrication and Fulfillment process.

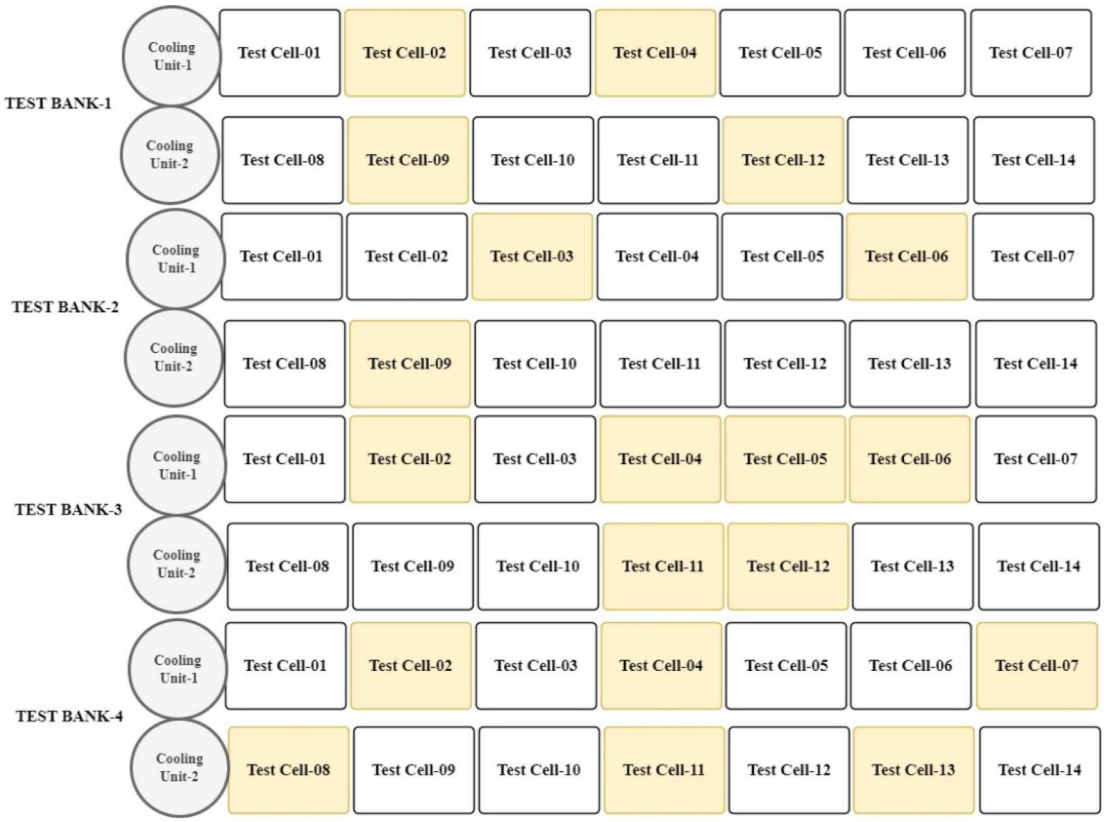


Fig. 2. Testing facility layout with four test banks, each containing fourteen test cells (yellow for idle, white for active) and two cooling units (gray circles).

we observed that tighter due dates increase the number of tardy servers.

The subsequent sections of this article are organized as follows. Section 2 provides a concise review of relevant literature. Section 3 presents our proposed mathematical model, while Section 4 summarizes the proposed solution methodology. Section 5 details our experimental setup and derives insights from the results. Finally, Section 6 summarizes our findings and provides recommendations for potential future extensions.

2. Literature review

The online resource allocation in manufacturing is vast and includes aspects like online scheduling, dynamic scheduling, online generalized assignment, dynamic assignment, and online packing problems (Balseiro, Kroer, & Kumar, 2023; Bukkur, Shukri, & Elmardi, 2018; Herrmann, 2006; Jaillet & Lu, 2011; Lu, 2013; Pinedo, 2012). In the following sections, we summarize research pertinent to our work, specifically around online scheduling and bin packing variants.

Research in online scheduling in manufacturing covers a range of topics, focusing on resource allocation, reactive scheduling, and optimization techniques. Gupta and Palis (2001) explored online real-time preemptive scheduling on multiple machines, developing algorithms to adhere to deadlines and optimize resource utilization. Mezmaz et al. (2011) introduced a parallel bi-objective hybrid metaheuristic for energy-efficient scheduling in cloud computing, aiming to balance energy consumption and performance. Cheng et al. (2013) proposed an energy-aware resource service scheduling approach for cloud manufacturing systems, integrating utility models to optimize resource use and minimize energy consumption, demonstrating enhanced efficiency and resource sharing in cloud-based decentralized manufacturing. Zhang, Wang, Liu, and Qian (2017) proposed a game theory-based strategy for real-time shop floor scheduling in cloud manufacturing, using a novel allocation strategy and dynamic optimization method to enhance processing efficiency. Kocsi, Matonya, Pusztai, and Budai (2020) designed a real-time decision-support system for scheduling in high-mix low-volume production within Industry 4.0, blending mathematical optimization with a genetic algorithm to analyze production risks and optimize scheduling. Wang, Wang, and Xu (2022) focused on simultaneous production and maintenance scheduling in refinery processes, optimizing scheduling decisions with attention to risk management and resource availability.

Online bin packing has been extensively studied in the literature due to its practical applications in various domains, including logistics, resource allocation, and scheduling. Table 1 summarizes the literature related to online bin packing and its variants. Seiden (2002) introduced a framework for analyzing online bin packing algorithms, enhancing algorithmic efficiency. Böhm, Sgall, van Stee, and Veselý (2017) focused on online bin stretching with three bins, applicable in server upgrades and shipment checking. Gupta and Radovanović (2020) proposed primal–dual algorithms for stochastic bin packing, with applications in shipping logistics and appointment scheduling. Bódis and Balogh (2019) explored an online variant with scenarios, applicable in diverse production lines. Angelopoulos, Kamali, and Shadkami (2023) addressed variants of the online bin packing problem with erroneous predictions, proposing efficient solutions for dynamic resource management tasks like virtual machine placement for server consolidation and memory allocation in data centers. Similarly, Zhao, She, Zhu, Yang, and Xu (2021) also focused on a variant of the online bin packing problem, specifically with 3D constraints, offering solutions applicable in logistics, manufacturing, and warehousing. Zhao, Zhu, Xu, Huang, and Xu (2022) conceptualized the online 3D bin packing problem as a Markov Decision Process, solved using deep reinforcement learning and highlighted its application in autonomous packing for logistics hubs and manufacturing plants. Boyar et al. (2001a, 2001b) and Epstein and Favrholdt (2003) examined online dual bin packing and its variants, focusing on resource allocation and optimization in online environments.

Dynamic bin packing, and its online variants, address the dynamic

Table 1
Summary of the research studies on online bin packing and its variants.

Online bin packing	Dynamic bin packing	Online dynamic bin packing	Online dual bin packing	Dynamic dual bin packing	Online bin packing with lookahead	Online dual bin packing with lookahead	Online dynamic dual bin packing with lookahead
O-BP	D-BP	OD-BP	O-DBP	D-DBP	O-BP-LA	O-DBP-LA	OD-DBP-LA

Reference	Variant								Research focus and methodology			Solution approach		Application domain
	O-BP	D-BP	OD-BP	O-DBP	D-DBP	O-BP-LA	OD-BP-LA	OD-DBP-LA	Analysis	Optimization	Machine Learning	Exact	Algorithm	
Zhao et al. (2021), Zhao et al. (2022)	X										X		X	General mfg, Logistics
Ojha et al. (2021)						X			X	X			X	Automated packaging systems
Boyar et al. (2021)							X		X				X	
Seiden (2002), Renault et al. (2015), Böhm et al. (2017), Gupta and Radovanović (2020), Angelopoulos et al. (2023), Bódis and Balogh (2019)	X								X				X	General mfg.,Data center operations, IT
Dunke and Nickel (2016), Polyakovskiy and M'Hallah (2018)						X				X		X	X	General mfg.,Data center operations
Boyar et al. (2016), Angelopoulos et al. (2018), Dunke and Nickel (2021)						X			X				X	Logistics, Transportation
Li et al. (2014), Buchbinder et al. (2021), Murhekar et al. (2023)		X							X				X	Data center operations, General mfg., Cloud computing service
Burcea (2014), Berndt et al. (2020), Guruganesh (2018)			X						X				X	General mfg., Cloud computing service
Boyar et al. (2001a, 2001b), Epstein and Favrholdt (2003)				X					X				X	General mfg., data center operations
Runarsson et al. (1996)					X								X	General mfg.
Our contribution								**X**		**X**		**X**	**X**	**Server mfg.**

arrival and departure of items in real-world applications like warehouse and computer storage. Burcea (2014) focused on minimizing bin usage over time without item migration, while Berndt, Jansen, and Klein (2020) introduced a method to balance bin numbers and migration factors, adding a technique for small items. Li, Tang, and Cai (2014) applied these concepts to cloud resource allocation in cloud gaming through the MinTotal DBP algorithm. Guruganesh (2018) developed a dynamic bin packing algorithm with recourse for real-time applications in cloud computing, and Buchbinder, Fairstein, Mellou, Menache, and Naor (2021) enhanced this approach using machine learning for Virtual Machine assignments. Murhekar, Arbour, Mai, and Rao (2023) explored MinUsageTime DBP for allocating multi-dimensional, resource-intensive online jobs to cloud servers, analyzing algorithm performance with synthetic data.

Literature on using the lookahead to access the future information in online bin packing has received growing attention recently. Dunke and Nickel (2016) included online bin packing with lookahead, demonstrating its positive impact on reducing bin usage. Renault, Rosén, and van Stee (2015) and Boyar, Kamali, Larsen, and López-Ortiz (2016) explored online bin packing algorithms with advice, where the online algorithm receives bits of future information with each input, enabling decisions based on both current and future information. Boyar, Favrholdt, Kamali, and Larsen (2021) further investigated the bin covering problem within the advice framework, revealing that optimal solutions depend on specific advice sizes, with larger sizes offering marginal performance improvements. Polyakovskiy and M'Hallah (2018) introduced a hybrid approach to the two-dimensional bin packing problem with lookahead, integrating heuristics and lookahead strategies, adaptable to various complex packing challenges. Angelopoulos, Dürr, Kamali, Renault, and Rosén (2018) explored the advice complexity of online bin packing, showing how small advice enhances algorithm performance, with simpler algorithms converging quicker than complex ones. Ojha et al. (2021) addressed the online 3D bin packing problem in automated robotic sorting centers, introducing algorithms and a framework to optimize bin packing heuristics using lookahead information, supported by comparative analyses with synthetic and industry data. Dunke and Nickel (2021) presented an exact analysis of online bin packing algorithms with and without lookahead, highlighting the advantages of lookahead and analyzing performance ratios in specific scenarios, providing detailed insights into the impact of lookahead on these algorithms.

Our review of the above literature reveals the following gaps that limit us from addressing the questions we raised in Section 1:

- There are limited studies in online and dynamic assignment considering effect of changes in lookahead window on the resulting assignment and decision-making process.
- Existing literature on online bin packing approaches does not consider simultaneously various real-world requirements such as server-test cell compatibility, test cell requirement, variable capacity of testing facility, and due date priority to maximize the number of servers tested.

To address these gaps, we propose an optimization model and a decomposition algorithm for the OD-DBP-LA, which we detail below.

3. Proposed optimization approach

The primary purpose of our proposed model for OD-DBP-LA is to facilitate online and dynamic assignment during the fulfilment test process in computer server industry. The input elements are revealed based on the time-lookahead approach and tested based on the parallel-random-order processing approach, enabling the concurrent processing of multiple elements and flexibility to process any available unfinished input at any time.

As indicated in Section 1, we take a dual bin packing approach whereby the test cells are bins, each preassigned with fixed attributes such as voltage and available time during the planning horizon. Further, servers are 'items,' each characterized by distinct voltage, number of test cells it requires for testing, and processing time needs. To elaborate, customer orders (servers) have varying voltage (high or low) and cooling (air or water) requirements, which are tested on dedicated test cells with specific capabilities. Each test cell has either high or low voltage power supply, or both. Additionally, while all test cells have air-cooling capacity, not all of them have water cooling capacity. The primary objective of the model is to minimize the number of unfinished servers, while ensuring that servers with earlier due dates are prioritized.

With this background, we now present an integer linear programming model for this OD-DBP-LA problem under the following assumptions:

- Each test cell can handle one server at a time.
- A server can be assigned to either one or two adjacent test cells simultaneously (i.e., 2TC) based on the number of cell requirements.
- No preemption is allowed; i.e., once a server is assigned to a test cell, it cannot be removed until its testing is completed.
- A test cell is immediately made available after testing of an assigned server is complete.
- All time-related aspects (arrival, processing, and due) are known with certainty at the beginning of a lookahead window and are deterministic.
- There are no delays due to breakdowns, failures, machine setup, unavailability of operators, changeovers, or any other reasons.

The model parameters and decision variables are listed in Tables 2 and 3, respectively, where parameters represented in bold denote a binary parameter matrix. Before we introduce the mathematical model, we first discuss the lookahead concept and the penalty function.

3.1. Lookahead window

The lookahead window (w) is defined as the number of future periods for which information is available to the decision-makers with certainty. In our model, it can be calculated as the difference between the start (S_m) and end (E_m) time of the window for a specific window m, inclusively ($w = E_m - S_m + 1$), where $w \ll L$ and L is the total length of the planning horizon. Therefore, at any given time t (where $t = S_m$), the decision-makers have access to all available information within the window $[t, t+w)$, where $E_m = t + w - 1$. Note that these lookahead windows are sequential and non-overlapping. Fig. 3 presents an illustration of three such lookahead windows, each of length w. All servers, both input and output, are represented by dropdown arrows, with different colors used to distinguish between them. Input servers are shown in light gray, while output servers are color-coded to represent their status; i.e., green for servers fully assigned, blue for partially assigned servers, and orange for those that remain unassigned. Servers that are partially assigned or unassigned in one window (such as window 1) are carried over to subsequent window(s) while the initial assignment position of partially assigned servers remains fixed until their testing is finished. These carried-over servers, along with any new arrivals, are treated as inputs for the next window (window 2), and this process is repeated for the rest of the windows until the end of the planning horizon.

Table 2
Parameters in the model.

Notation	Definition
I	Set of test banks; $i \in I$
J	Set of test cells; $j \in J$
K	Set of servers; $k \in K$
L	Length of the planning horizon
w	Length of a lookahead window
M	Set of lookahead window(s); $m \in M = \left\{1, 2, \cdots, \left\lceil \frac{L}{w} \right\rceil \right\}$
S_m	Start time of a lookahead window $= (m-1)\,w + 1$
E_m	End time of a lookahead window $= min\{mw, L\}$
T_m	Set of discretized time period(s) for a specific window; $t \in T_m$ and $\lvert T_m \rvert = w$
D_k	1, if server k requires 2 test cells (2TC); 0, otherwise
H_K	1, if server k requires high voltage supply; 0, otherwise.
L_K	1, if server k requires low voltage supply; 0, otherwise
W_K	1, if server k requires water cooling; 0, otherwise
$\boldsymbol{H_{ij}}$	1, if j^{th} test cell under i^{th} test bank has high voltage power supply available; 0, otherwise
$\boldsymbol{L_{ij}}$	1, if j^{th} test cell under i^{th} test bank has low voltage supply available; 0, otherwise
$\boldsymbol{W_{ij}}$	1, if j^{th} test cell under i^{th} test bank has water cooling unit available; 0, otherwise
$\boldsymbol{A_{kt}}$	1, if server k is available at time t; 0, otherwise
P_k	Test processing time required for each server k
C_k	Cost associated with each server k
A_k	First period when a server k is available, $A_k = \min\{t, \forall t \in T_m \mid \boldsymbol{A_{kt}}=1\}$; $\forall k \in K$
N	Set of blocking instances for other testing processes (e.g., fabrication) in the same testing facility; $n \in N$
i_n	Blocked test bank due to other testing processes
j_n	Blocked test cell due to other testing processes
ST_n, ET_n	Blocking start and end times for other testing processes
Z	Set of instances for carried-over servers with remaining time; $z \in Z$
O_z	Carried-over server with remaining time
B_z	Bank holding carried-over server for the remaining time
C_z	Test cell(s) holding carried-over server for the remaining time
R_z	Remaining time for carried-over server
N_i	Test bank(s) entirely unavailable throughout the planning horizon
N_j	Test cell(s) entirely unavailable throughout the planning horizon

Table 3
Decision variables in the model.

Notation	Definition
x_{kijt}	1 if a server k is assigned to j^{th} test cell under i^{th} test bank at time t; 0, otherwise
y_k	1 if a server k is assigned; 0, otherwise
b_{kij}	1 if a server k is assigned to j^{th} test cell under i^{th} test bank; 0, otherwise
a_{kt}	1 if a server k is starting at a particular time t; 0, otherwise
o_{kij}, c_{kt}	auxiliary binary variables, {0,1}
f_k	Time-period when server k is first assigned in window w

3.2. The dynamic penalty function

The optimization model minimizes the number of unfinished servers where we prioritize unassigned servers (in each lookahead window) based on the earliest due date. To do so, we use a dynamic penalty cost function, $C_k = (q-i)\Psi$, where C_k represents the cost assigns to a particular server, q is the total number of servers for a specific window, i indicates the server's position (index) in the sorted list, and Ψ is a very large number. That is, the cost function assigns a higher cost to servers with earliest due date and lower cost to those with latest due date in a linear manner. When multiple servers share the same due date, the server that arrived earlier receives a higher cost. Servers with identical arrival and due dates are assigned costs based on their numerical order. Table 4 provides an example of 5 servers with their arrival and due times, and Table 5 provides their cost assignment for the sorted server list, considering $\Psi = 100{,}000$. Note that the assignment of servers partially assigned in the previous window remains unchanged in the current window being solved.

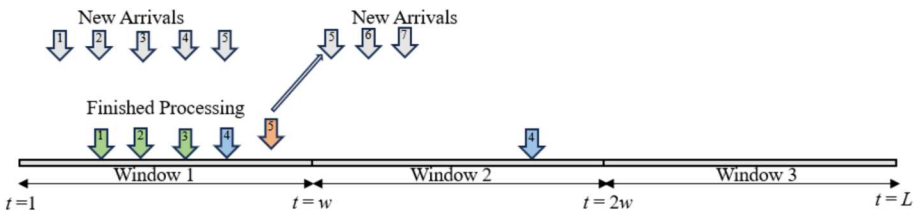


Fig. 3. Illustration of the lookahead concept with server assignment (light gray for input, green for fully assigned, blue for partially assigned, and orange for unassigned).

3.3. Proposed model

Given these preliminaries, we now present our proposed mathematical model for OD-DBP-LA.

Minimize $\sum_k C_k(1-y_k)$

Subject to:

$$\sum_k x_{kijt} \le 1; \quad \forall i \in I, j \in J, t \in T_m \tag{1}$$

$$\sum_i \sum_j x_{kijt} \le 1 + D_k; \quad \forall k \in K, t \in T_m \tag{2}$$

$$\sum_j x_{kijt} \le 1 + D_k; \quad \forall i \in I, k \in K, t \in T_m \tag{3}$$

$$\sum_t \sum_k x_{kijt} = 0; \quad \forall i \in N_i, \forall j \in N_j \tag{4}$$

$$\sum_t \sum_i \sum_j x_{kijt} \boldsymbol{A_{kt}} = P_k(1+D_k)\, y_k; \quad \forall k \in K \tag{5}$$

$$\sum_t^{A_k-1} \sum_i \sum_j x_{kijt} = 0; \quad \forall k \in K \tag{6}$$

$$\sum_i \sum_j b_{kij} = 1 + D_k; \quad \forall k \in K \tag{7}$$

$$o_{kij} \le b_{kij}; \quad \forall k \in K, i \in I, j \in J \backslash \{\max(J)\} \tag{8}$$

$$o_{kij} \le b_{ki(j+1)}; \quad \forall k \in K, i \in I, j \in J \backslash \{\max(J)\} \tag{9}$$

$$o_{kij} \ge b_{kij} + b_{ki(j+1)} - 1; \quad \forall k \in K, i \in I, j \in J \backslash \{\max(J)\} \tag{10}$$

$$\sum_i \sum_j^{J\backslash\{\max(J)\}} o_{kij} = D_k; \quad \forall k \in K \tag{11}$$

$$x_{kijt} \le b_{kij}; \quad \forall k \in K, i \in I, j \in J, t \in T_m \tag{12}$$

$$\sum_t a_{kt} = 1; \quad \forall k \in K \tag{13}$$

$$f_k = \sum_t t a_{kt}; \quad \forall k \in K \tag{14}$$

$$\sum_{t=S_m}^{f_k-1} c_{kt} = 0; \quad \forall k \in K, t \in T_m, m \in M | f_k \ne 1, a_{kt} = 1 \tag{15}$$

$$\sum_{t=f_k}^{\min\{f_k+P_k-1, E_m\}} c_{kt} = \min\{P_k, E_m - f_k + 1\}; \quad \forall k \in K, t \in T_m, m \in M | a_{kt} = 1 \tag{16}$$

Table 4
Server arrivals for a window.

Server (k)	Arrival Time (Days)	Due Date (Days)
1	1	5
2	2	5
3	2	2
4	3	6
5	4	2

Table 5
Cost of sorted servers.

Sorted servers (k)	Cost (C_k)
3	500,000
5	400,000
1	300,000
2	200,000
4	100,000

$$\sum_{t=f_k+P_k}^{E_m} c_{kt} = 0; \quad \forall k \in K, t \in T_m, m \in M | f_k + P_k \le E_m, a_{kt} = 1 \tag{17}$$

$$x_{kijt} \le c_{kt}; \quad \forall k \in K, i \in I, j \in J, t \in T_m \tag{18}$$

$$H_{ij} - H_k - x_{kijt} \ge -1; \quad \forall k \in K, i \in I, j \in J, t \in T_m \tag{19}$$

$$L_{ij} - L_k - x_{kijt} \ge -1; \quad \forall k \in K, i \in I, \mathrm{j} \in J, t \in T_m \tag{20}$$

$$W_{ij} - W_k - x_{kijt} \ge -1; \quad \forall k \in K, i \in I, j \in J, t \in T_m \tag{21}$$

$$\sum_{t=\max\{ST_n, S_m\}}^{\min\{ET_n, E_m\}} x_{ki_nj_nt} = 0; \quad \forall k \in K, \forall n \in N, m \in M \tag{22}$$

$$\sum_{t=S_m}^{t=S_m+\min\{w, R_z\}-1} x_{O_zB_zC_zt} = \min\{w, R_z\}; \forall \mathrm{z} \in \mathrm{Z}, m \in M \tag{23}$$

$$x_{kijt} \in \{0,1\}; \quad \forall k \in K, i \in I, j \in J, t \in T_m \tag{24}$$

$$y_k \in \{0,1\}; \quad \forall k \in K \tag{25}$$

$$b_{kij} \in \{0,1\}; \quad \forall k \in K, i \in I, j \in J \tag{26}$$

$$a_{kt} \in \{0,1\}; \quad \forall k \in K, t \in T_m \tag{27}$$

$$o_{kij} \in \{0,1\}; \quad \forall k \in K, i \in I, j \in J \tag{28}$$

$$c_{kt} \in \{0,1\}; \quad \forall k \in K, t \in T_m \tag{29}$$

$$f_k \in \mathbb{Z}^+ \cap T_m; \quad \forall k \in K \tag{30}$$

Note that this model needs to be solved iteratively for all the windows, $m \in M$, for a specific length of lookahead window, w. So, if we define the set of lookahead window lengths as $W = \{1, 2, 3, \ldots\ldots, L\}$, where L is the total length of the time horizon, then each specific lookahead window is represented as $LA_{w,m}$. Here $LA_{w,m}$ represent the m^{th} window for a lookahead (LA) window of length w.

Constraints (1) enforce the capacity limit for test cells at a specific time, stating that each test cell in a test bank can handle only one server at a time. Constraints (2) and (3) enforce assignment restrictions for servers, ensuring that they cannot be assigned to more than two test cells in a test bank at a time. Constraints (4) restrict the assignment of servers to test cells that are completely unavailable throughout the entire planning horizon. Constraints (5) specify a limit on the testing time for servers in test cells. It ensures that if a server k is available at a given time t and assigned to 1 test cell (1TC) or 2 test cells (2TC) inside a test bank, the total time it spends in that test cell will not exceed the server's total testing time requirement (P_k). Constraints (6) restrict server assignment

before arrival. It specifies that for any server k, it cannot be assigned to any test cell(s) j under any test bank i when it's not available in the system. Some servers require 2TC that must be adjacent to each other and located under a single bank. Constraints (7) ensure that a server k will be assigned to 1TC if $D_k = 0$ or 2TC if $D_k = 1$, regardless of the time. Constraints (8)-(12) specify that if a server requires 2TC ($D_k = 1$), those cells should be adjacent to each other under one bank i. It is important to note that to maintain this adjacency requirement, we exclude the last test cell denoted as max(J), within a test bank from consideration, as it is not feasible to assign a server (that requires 2TC) to the last cell in a test bank.

Constraints (13)-(16) ensure the assignment of servers to test cells for contiguous testing times. Constraints (13) ensure that each server k is assigned exactly one start time ($a_{kt} = 1$) and Constraints (14) utilize the variable f_k to capture that specific start time for each server k. Constraints (15) ensure that a server cannot be assigned before its specified starting time of a window, except for servers that commence at the beginning of the window. Constraints (16) enforce that a server cannot be assigned any time before its assignment, aligning the processing duration with either the server's testing duration or the remaining time in the window. Constraints (17) ensure that no server is assigned beyond the time period required for its testing completion, specifically focusing on scenarios when the testing finishes within the current window. Constraints (18) ensure that if a server cannot be assigned to any time-period, it cannot be assigned to any bank or cell for the entire planning horizon. Constraints (19)-(21) govern the assignment based on voltage (high and low) and water-cooling compatibility. These constraints ensure that if a server requires high voltage (H_k), it cannot be assigned to a cell without high voltage ($H_{ij} = 0$), and similarly for low voltage (L_k) and water cooling (W_k).

Constraints (22) capture the unavailability of test cells that are occupied with other processes (e.g., fabrication) and gradually become available as the time window progresses. This ensures that no servers will be assigned to these preoccupied test cells until they become available. Constraints (23) govern the carried-over assignment of servers O_z across lookahead windows based on their remaining testing time R_z. It specifies that any server with remaining testing time must be assigned to the same test bank and cell to complete their testing. Constraints (24)-(30) define bound on decision variables.

Note that our proposed ILP model, a variant of the classical bin packing problem, is NP-hard (Parikh & Meller, 2008; Vijayakumar et al., 2013). Consequently, solving realistic problem instances can be challenging. Our preliminary experiments indicated that state-of-the-art commercial software like CPLEX, although effective for smaller instances, faces difficulties with longer lookahead windows in large scenarios. Specifically, in industry-scale instances involving 400 + servers with a minimum of 20 % 2TC requirements for longer lookahead windows, CPLEX fails to deliver optimal solutions even with more than 24 h of runtime, making it an impractical choice for solving the proposed model. To overcome the limitations of solving the model with a commercial solver, we propose a '2-phase' computational framework that integrates mathematical programming with a genetic algorithm (GA). We will now discuss our proposed approach in detail.

4. A 2-phase computational framework

Recall that there are two types of servers, those requiring a single test cell (1TC) and those requiring two (2TC), which must also be adjacent to each other. The presence of constraints related to 2TC servers complicate the problem. Fig. 4 demonstrates the relationship between runtime and solution gap with the percentage of servers requiring 2TC when solving a problem with 300 servers and 54 test cells for lookahead window length of 10 with a state-of-the-art commercial solver. This challenge is more pronounced in larger instances with longer lookahead windows.

To mitigate this complexity in the OD-DBP-LA, we propose a two-phase computational framework that seamlessly integrates mathematical programming with GA for a given lookahead window m:

- **Phase 1:** This phase uses GA to solve the original problem of assigning all servers (with 1TC and 2TC requirements); recall, the original problem cannot be solved efficiently using a commercial solver. With GA having full visibility into all servers, it provides an efficient first-cut solution to the original problem.

Algorithm 1: '2-phase' Computational Framework

Define the parameters W_{ij}, H_{ij}, L_{ij}, L
For $w \in W$ **do**:
  Calculate the lookahead window set, $M = \{1,2,3,\ldots,\left\lceil \frac{L}{w} \right\rceil\}$
  For $m \in M$ **do**:
    Define parameters S_m, E_m, T_m, K, P_k, C_k, W_k, H_k, L_k, D_k, R_z, $\boldsymbol{A_{kt}}$
    IF ($K! = Null$):
      Solution_Phase1 = GA(w, m, S_m, E_m, T_m, K, P_k, C_k, W_k, H_k, L_k, D_k, R_z, $\boldsymbol{A_{kt}}$, $\boldsymbol{W_{ij}}$, $\boldsymbol{H_{ij}}$, $\boldsymbol{L_{ij}}$)
      Solution_Phase2 = MathModel(w, m, S_m, E_m, T_m, K, P_k, C_k, W_k, H_k, L_k, D_k, R_z, $\boldsymbol{A_{kt}}$, $\boldsymbol{W_{ij}}$, $\boldsymbol{H_{ij}}$, $\boldsymbol{L_{ij}}$, Solution_Phase1)
      Extract data from Solution_Phase2
      Calculate the assigned time for each server and pass servers with the remaining time and those unassigned to the next iteration or window (m)
      Save the data
    ELSE:
      Continue to the window (m)
    EndFor
EndFor

- **Phase 2:** In this phase, the 2TC assignments from the solution in Phase 1 are fixed and then the reduced problem is resolved using a commercial solver. The idea is to leverage the mathematical programming approach to derive an optimal solution to the 1TC problem by removing 2TC decisions from consideration (as they were already assigned in Phase 1). This combination of metaheuristic and mathematical programming approach, solved in a sequential manner, allowed us to achieve high-quality solutions in a reasonable amount of time (see Section 5). Algorithm 1 provides a pseudo-code of the proposed '2-phase' computational framework. More details about the proposed framework are available in the Appendix. We now discuss the two phases in detail.

4.1. Phase 1 (genetic algorithm)

Our GA implementation in Phase 1 to assign both server types (1TC

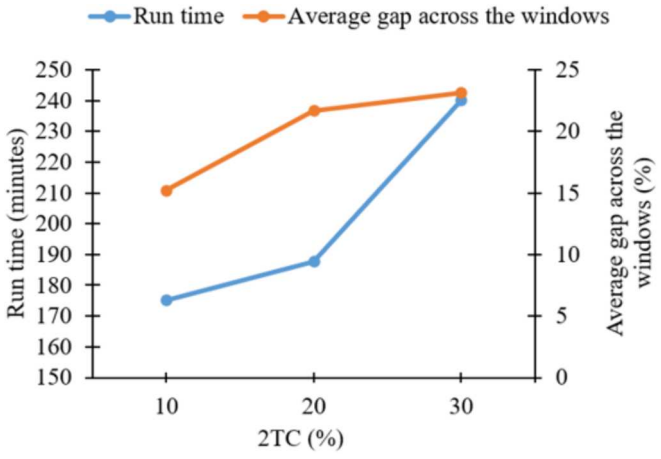


Fig. 4. Run time and gap with varying 2TC.

and 2TC) in a specific lookahead window is detailed below.

4.1.1. Solution representation

Each chromosome in the population is represented as a vector of vectors and matrices, where each gene represents a server. The primary structure of the chromosome is a vector, which is further divided into $|I|$ equal-sized vectors, where each vector (φ_i) corresponds to a test bank i. Each vector (φ_i) for a bank i is further subdivided into $|J|$ vectors of equal size, where each vector (φ_{ij}) corresponds to a test cell j, arranged in sequence. The sequence of servers within the resulting matrix (φ_{ij}) for a test cell j represents the operational sequence for that cell j under bank i. The server listed first in matrix (φ_{ij}) will be processed first and will occupy cell j from the earliest available time up to its processing time, calculated based on the length of lookahead window (w) and its arrival time. If a server cannot be processed, then it will be considered unassigned.

Fig. 5 represents the nested vector–matrix structure of the chromosome for a system, where (i) $I = \{1,2\}$, (ii) $J = \{1,2,3\}$, (iii) $K = \{G_1, G_2, \ldots, G_{12}\}$; i.e., 12 servers, and (iv) window length $w = 5$. Note that we only illustrate the example for bank, $i = 2$ to illustrate the subsequent structure after dividing chromosome into sub-sections correspond to each bank i.

In the final step **(**Time allocation for each element in φ_{ij}**),** within vector $\boldsymbol{\varphi}_{22}$, gene $\boldsymbol{G_9}$ appears first and occupies test cell 2 (under bank 2) for time 1, 2 and 3; the 3 time periods correspond to the processing time for server $\boldsymbol{G_9}$. Subsequently, gene $\boldsymbol{G_{10}}$ is processed occupying test cell 2 under test bank 2 for time 4 and 5.

4.1.2. GA cycle

The GA cycle consists of the following stages, culminating in a set of termination criteria:

1. **Initial Population:** Generate initial population (chromosomes) and evaluate the population.
2. **Selection:** Parents for mating are selected using a parent selection operator.
3. **Crossover:** Mating occurs after parent selection. Parents are combined into one or multiple offspring by the crossover operator.
4. **Mutation:** After offspring are created in the crossover stage, mutation is carried out using a mutation operator with a predefined probability. This is done to increase diversity in the population.
5. **Evaluation:** The evaluation is done in two steps. Initially, the validity of each chromosome in the initial population is checked to ensure they satisfy the constraints. Subsequently, the objective or fitness value is calculated for valid solutions.
6. **Survival**: Chromosomes are sorted based on their fitness value.
7. **Termination:** GA will terminate when any of the following termination criterion are met: (i) after 4,000 generations or (ii) no improvement for 100 consecutive generations.

If the above termination criteria are not met, then the process will go back to step 2 (selection).

4.2. Phase 2 (mathematical programming)

In Phase 2, we utilize the mathematical programming approach to find the optimal solution for 1TC servers. To do this, using the solution from Phase 1, we first fix the assignment of servers requiring 2TC for the time they were assigned in Phase 1 by adding the following constraint to the original mathematical model:

$$\sum_{t=\rho_k}^{\tau_k} (x_{ki_kj_kt} + x_{ki_k(j_k+1)t}) = 2(\tau_k - \rho_k + 1); \quad \forall k \in G, \tag{31}$$

where $G \subseteq K$ is assigned in Phase 1. Each server k requiring 2TC should be allocated to its designated test bank i_k and corresponding cell or cells (j_k) from the start (ρ_k) to end (τ_k) time period it was assigned in Phase 1. Fig. 16 in the Appendix summarizes the key steps for our '2-phase' computational framework.

Second, to ensure that unassigned servers with 2TC in Phase 1 are excluded, we introduce an additional constraint in the model. This constraint ensures that such servers not assigned in Phase 1 are also not considered in the current solution of Phase 2.

$$x_{kijt} = 0; \quad \forall k \in U, i \in I, j \in J, t \in T_m, \tag{32}$$

where $U \subseteq K$ with 2TC in Phase 1, $k \in U$. If a server requiring 2TC is not assigned in Phase 1, it should not be considered for assignment in any

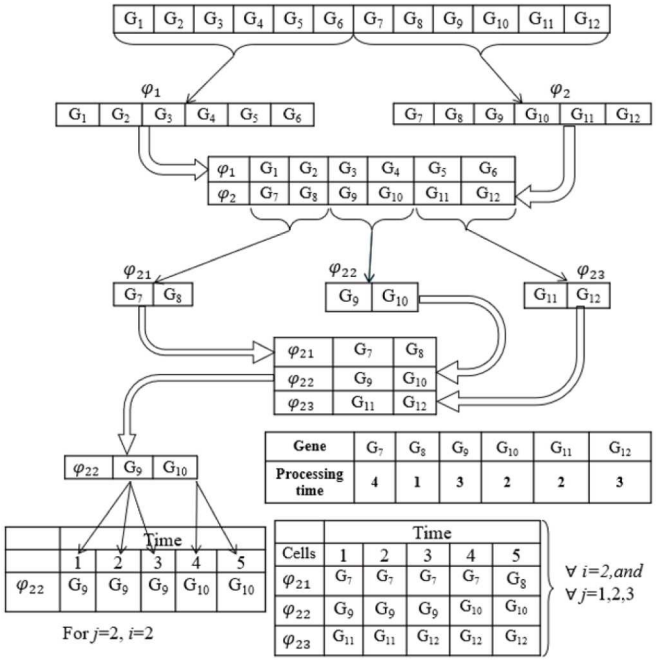


Fig. 5. Solution encoding.

test bank or cell(s) during the same lookahead window in Phase 2 either. The resulting mathematical model (original model plus Constraints (31) and (32)), is solved using a commercial solver to optimize the assignment of servers requiring single cells.

In Phase 1, GA was implemented using the Pymoo optimization framework, while in Phase 2, we utilized DOcplex, a native Python optimization library. The algorithms for both stages were coded in Python version 3.9 and run on a Dell I7-10700 CPU @ 2.90 GHz desktop with 32 GB of RAM to find the solutions.

5. Computational study

We now present our computational study starting with data generation and GA parameter tuning. We then evaluate our solution approach, conduct a sensitivity analysis, and conclude with key insights.

5.1. Data generation and GA parameter tuning

The testing facility that we select is similar to our industry partner's site as illustrated in Fig. 2. We consider a time horizon of 13 weeks, corresponding to 1 quarter in a year. Fig. 7 depicts the availability of test cells for testing considering that these cells are often allocated to other process in the manufacturing facility during the earlier part of the planning horizon.

Because the data available from our industry partner was limited, we extrapolated this to generate a synthetic dataset. Two general types of arrival patterns are typically encountered in a server manufacturing industry: 'left-skewed' and 'right-skewed.' As the names suggest, the 'right-skewed' pattern would have most servers arriving during the earlier part of the planning horizon, which is opposite of a 'left-skewed' pattern where most servers arrive during the last part. We also considered 'uniformly loaded' arrival patterns for comparison purposes. Fig. 6 illustrates the monthly distribution of server arrivals for all three patterns using an example dataset of 400 arrivals over the entire planning horizon. The colors green, orange, and purple represent months 1, 2, and 3, respectively. Table 6 summarizes the % of servers in each month for each arrival pattern.

In terms of GA, we employed a systematic approach to determine the appropriate values for the GA's parameters. This involved an iterative tuning process using smaller instances of the original problem. This method allowed us to fine-tune the parameters effectively, and ensure robust and reproducible results. Specifically, in terms of population size, to balance genetic diversity and computational efficiency, we tested various population sizes. A population size of 100 provided the best balance, ensuring sufficient diversity without excessive computational cost. Sizes of 25 and 50 produced poor results, while 150, 200, and 300 offered no significant improvement (and resulted in higher computation time) over a size of 100.

For crossover and mutation parameters, we tested various crossover operators (SBX, single-point, two-point, order) and mutation operators (bit-flip, inversion, polynomial) with different rates, along with dynamic adjustments to both crossover and mutation rates. The combination of SBX with a 0.5 crossover rate and polynomial mutation with a 0.2 rate comparatively provided the best results, ensuring high-quality solutions and preventing the algorithm from getting stuck in local optima without significantly increasing computation time.

5.2. Performance of the 2-phase computational framework

For the performance evaluation of the '2-phase' computational framework, we analyzed ten problem instances within an existing testing facility, utilizing two distinct server arrival patterns: uniform and right-skewed. We deliberately avoided the left-skewed pattern, as similar problem instances in this category are comparatively easier to solve and often yield almost identical outcomes. In the uniform pattern, we examined three server sizes—200, 300, and 400—while for the right-skewed pattern, we focused on sizes 300 and 400, since the outcomes for left-skewed 200 were nearly the same as those for uniform 200. For each server size in both arrival patterns, we consistently applied a specific percentage of 2TC, reflecting the proportion of servers that require two compatible cells to be located adjacently. A uniform 10-day lookahead window length was maintained in all instances. This approach, emphasizing more challenging scenarios, allowed for an equitable and comprehensive comparison of the framework's performance across diverse scenarios. We set a CPU-time limit of 2 h for each lookahead window when solving the problem with the CPLEX optimization solver.

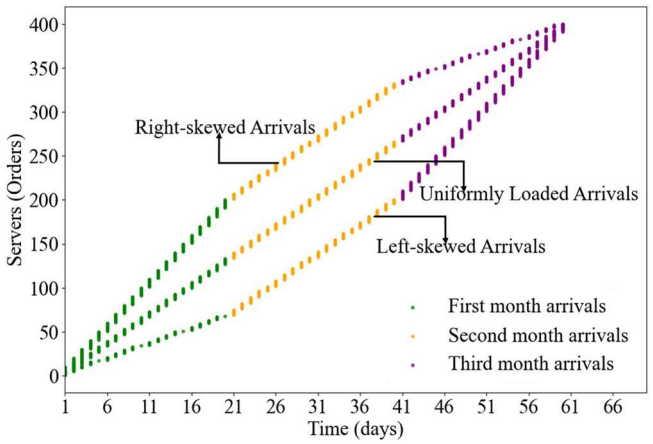


Fig. 6. Monthly distribution of server arrivals for right-skewed, uniformly loaded, and left-skewed patterns.

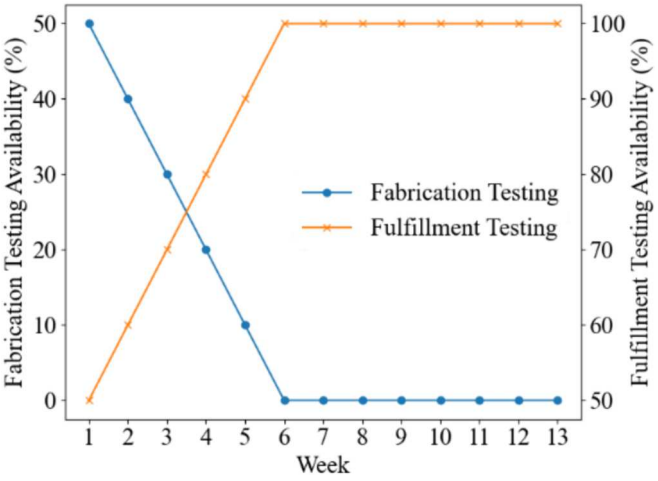


Fig. 7. Availability of test cells for fabrication and fulfillment testing over 13 weeks.

Table 6 presents our computational experiments that compare the solution quality and runtime of the '2-phase' computational framework and 'Exact approach' (per Section 3) for several problem instances. In the table, '% Difference' column represents the difference between the objective' i.e., % of the unfinished servers in the Exact approach and (%) of unfinished servers in '2-phase' computational framework. The positive value represents that the '2-phase' computational framework' outperformed the Exact approach by producing fewer unfinished servers than the Exact approach. The number in a bracket of the 'Exact' column of 'Solution Quality' represents the gap between the best solution and the lower bound when the solver reached the time limit.

These computational experiments confirm that our '2-phase' computational framework can achieve high quality solutions (often, optimal) across different scenarios in a relatively quick time; therefore, we utilized this approach for further experiments to generate insights.

Table 6
Performance evaluation of our proposed '2-phase' computational framework.

Experiments				Solution quality (number of unfinished servers)			Run time (minutes)	
Arrival pattern	Demand size	% of 2TC	Lookahead window (w)	Exact	2-phase	% Diff	Exact	2-phase
Uniform	200	10 %	10	0	0	0	0.05	2.38
		30 %	10	0	0	0	0.79	2.52
	300	10 %	10	0	0	0	0.39	4.09
		30 %	10	0	0	0	216.58	4.45
	400	10 %	10	0	0	0	325.86	92.09
		30 %	10	140 (39.0 %)	0	35 %	840.00	190.85
Right-skewed	300	10 %	10	0	0	0	244.66	63.42
		30 %	10	0	0	0	610.78	50.39
	400	10 %	10	165 (41.2 %)	0	41.25 %	840.00	287.23
		30 %	10	281 (32.1 %)	0	71.25 %	840.00	276.62

5.3. Experimental setting

Preliminary experiments indicated that solutions to the OD-DBP-LA problem were sensitive to system parameters; e.g., physical server allocation (1TC and 2TC), arrival patterns, server size, length of the lookahead window, and capacity utilization of the testing facility. Table 7 summarizes these factors, their levels, and values, with bold entries in the last column indicating the base case. The bounds on the input parameters were determined based on our interactions with the partnering industry.

Note that, while the primary objective in our model is the minimize the number of unfinished servers, since our model prioritizes servers with earlier due dates, we also calculated the number of tardy servers (servers completed beyond their due date) as a post-hoc measure.

5.4. Experimental insights

Insight 1: The effect of longer lookahead on reduction in unfinished servers diminishes at higher testing capacity and smaller server size.

In our approach, we assess the frequency of decision-making by examining the effects of 1-, 5-, and 10-day lookahead windows on outcomes at different testing capacities. Intuitively, a 10-day window would lead to a reduction in unfinished servers given increase in the window of certainty in terms of server arrivals, as shown in Fig. 8. This figure illustrates the impact on the number of unfinished servers (y-axis) with increasing test capacity (one, two, and four banks). The x-axis represents a combination of 2TC (%) and lookahead window length, with different colors representing the outcomes for the same 2TC (%). Also note that as testing capacity increases from 1 to 4 banks, the benefit of a longer lookahead window in terms of lower unfinished servers decreases.

Additionally, the effect of server size on the number of unfinished servers was analyzed. Fig. 9 illustrates the impact on the number of unfinished servers (y-axis) with decreasing server size (400, 300, and 200). The x-axis represents a combination of 2TC (%) and lookahead window length, with different colors representing the outcomes for the same 2TC (%). Also note that as server sizes decreases from 400 to 200 servers, the benefit of a longer lookahead window in terms of lower unfinished servers decreases. Smaller server sizes inherently require less capacity, enabling existing resources to efficiently handle demand variability and allocate resources effectively, thus reducing reliance on extended lookahead for optimizing outcomes.

Insight 2: As the lookahead window shortens, the variability in test processing times between servers and the proportion of servers requiring 2TC significantly affect the characteristics of unfinished servers.

We investigated the characteristics of unfinished servers and observed how they are affected by varying lookahead window lengths, leading to two main findings. First, with longer lookahead windows, servers requiring longer test processing times are more likely to be unfinished compared to those requiring shorter test processing times. In contrast, scenarios with shorter lookahead windows, and servers with shorter processing times are more frequently unfinished. This is because servers with shorter test processing times are often prioritized over servers with longer processing times (if due dates are comparable) to minimize the total number of unfinished servers when the lookahead window is longer. Notice the difference between the % of servers with longest processing time in the input data and output solution in Fig. 10; i.e., 27 % vs. 31.22 % for lookahead window of 10.

However, when the lookahead window length is decreased to less than the processing times of most servers, the capability to distinguish between servers based on their processing times diminishes. This results

Table 7
Summary of the parameters, levels, and values in the sensitivity analysis.

Parameters	Level	Values
Monthly (M) server arrival pattern (3 months)	**Right-skewed**	**M1: 50 %, M2: 33.33 %, M3: 16.67 %**
	Uniformly-loaded	M1: 33.33 %, M2: 33.33 %, M3: 33.33 %
	Left-skewed	M1: 16.67 %, M2: 33.33 %, M3: 50 %
2TC requirement	Low, Medium, High	**10 %**, 20 %, 30 %
Server size	Low, Medium, High	200, **300**, 400
Lookahead window length (days)	Short, Medium, Long	1, 5, **10**
Number of available banks (and test cells)	Low, Moderate, Full	1 (14 cells), 2 (28 cells), **4 (54 cells)**[1]
Due date flexibility (Average lead time)	Tight, Loose	19, **26**

[1] While each test bank typically has 14 cells, Test Bank 4 has 2 unusable cells, reducing the total available cells across all four banks to 54.

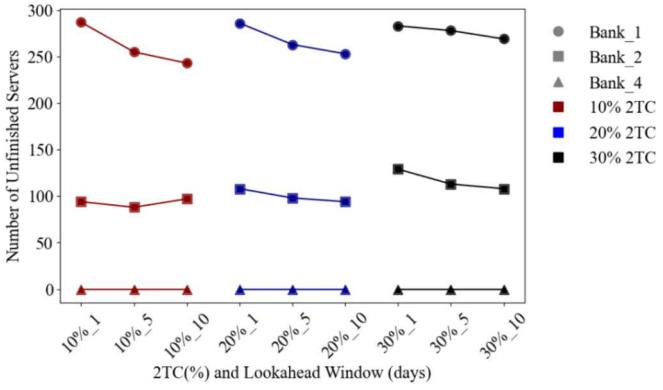


Fig. 8. Impact of test capacity on unfinished servers across 2TC (%) and lookahead windows.

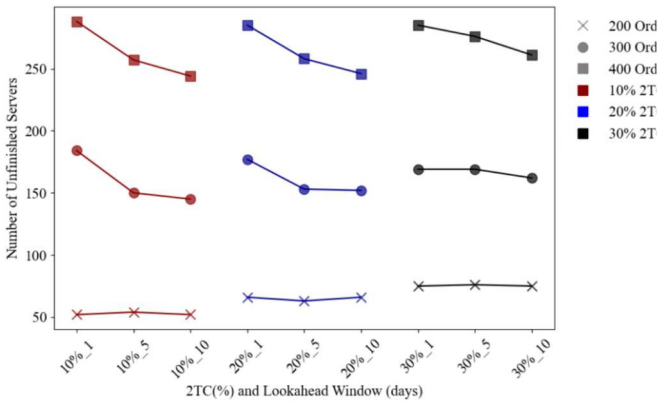


Fig. 9. Impact of server size on unfinished servers across 2TC (%) and lookahead windows.

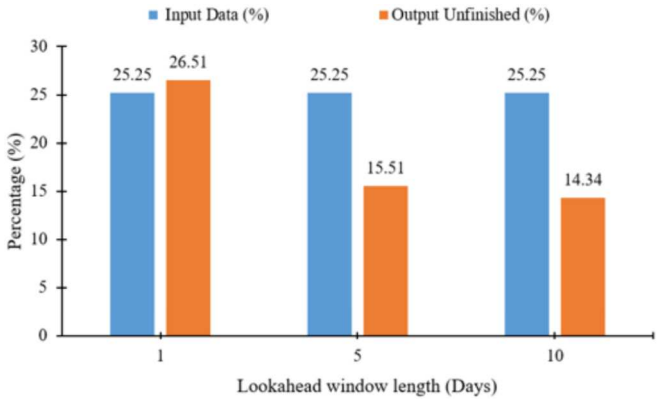


Fig. 11. Comparison of actual and unfinished percentages of short processing time servers across lookahead windows.

in a random selection of servers, which can increase the fraction of unfinished servers having longer processing times compared to the prior case when the lookahead is longer (i.e., 27 % vs. 12.5 % for lookahead window of 1). In contrast, Fig. 11 shows the opposite effect for servers with shorter processing times. For lookahead window of 1 day, a larger percentage of these servers remain unfinished (26.51 %), whereas at a lookahead window of 10 days, the unfinished rate decreases to 14.34 %, which is below the original proportion in the input data (25.25 %). With a lookahead window length of 5 days for both long and short test processing times, the unfinished rate aligns more closely with their representation in the data.

Second, we observe that with shorter lookahead windows, the proportion of unfinished servers requiring two test cells (2TC) tends to be significantly higher compared to their initial representation in the original data. Fig. 12 illustrates this trend, showing that with a 1-day lookahead window, the percentage of unfinished servers requiring 2TC is approximately 43.6 %. This percentage decreases slightly to 42.5 % with a 5-day window, and further to 41.4 % with a 10-day window. Notably, all these percentages are significantly higher than the actual presence of 2TC in the input data, which is around 30 %. A longer lookahead window, specifically for 10 days, provides a more comprehensive view of both 1TC and 2TC requirements over an extended period. This allows for effectively reserving resources for upcoming 2TC servers without compromising 1TC servers, while maintaining due date priority. Such an approach leads to a more balanced allocation of cells to both types of servers, thereby reducing the number of unfinished 2TC servers.

In contrast, a 1-day lookahead window may result in an allocation based on limited foresight. This limitation hinders the algorithm's ability to strategically reserve and align adjacent cells for 2TC assignments, resulting in a higher likelihood of a greater number of 2TC servers remaining unfinished compared to scenarios with longer lookahead windows.

Insight 3: Right-skewed arrival results in fewer unfinished servers, but more tardy servers; the effect is reversed with left-skewed arrival.

Fig. 13 demonstrates the variation in unfinished and tardy servers across different arrival patterns for a fixed 2TC (%) and lookahead window, with server sizes of 300 (lower lines) and 400 (upper lines), considering a system with 2 test banks. The number of tardy servers (servers completed beyond their due date) was calculated post-hoc. The vertical position of each bubble represents the number of unfinished servers, and the size of the bubble represents the number of tardy servers. We observe that with right-skewed arrivals, servers that arrive early have more time and capacity available to be finished, resulting in lower unfinished servers. However, the initial surge of servers inherent in this arrival pattern can overwhelm the testing facility, leading to delays in processing these servers leading more tardy servers; i.e., even though more serves get finished, many miss their due date. This effect is reversed for the left-skewed arrival pattern, where a mismatch in a large number of servers compared to available capacity in a shorter time-span results in many servers not even getting started causing a higher number of unfinished servers. Those servers that do get processed mostly get completed prior to their due date resulting in few tardy servers.

Insight 4: Tighter due dates result in a higher number of tardy servers without impacting the count of unfinished ones.

To assess the effect of tighter due dates, we created an additional dataset, keeping all other parameters constant. We ensured that each server had at least the minimum required time for test processing and

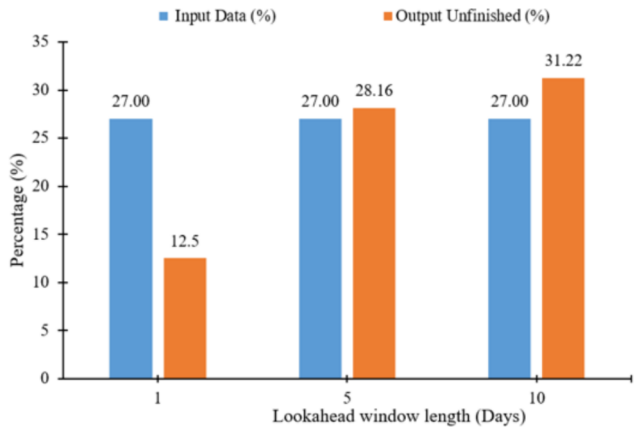


Fig. 10. Comparison of actual and unfinished percentages of long processing time servers across lookahead windows.

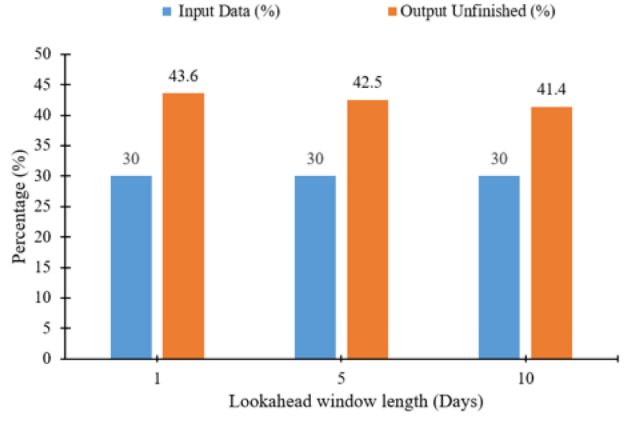


Fig. 12. Comparison of actual and unfinished percentages of 2TC servers across lookahead windows.

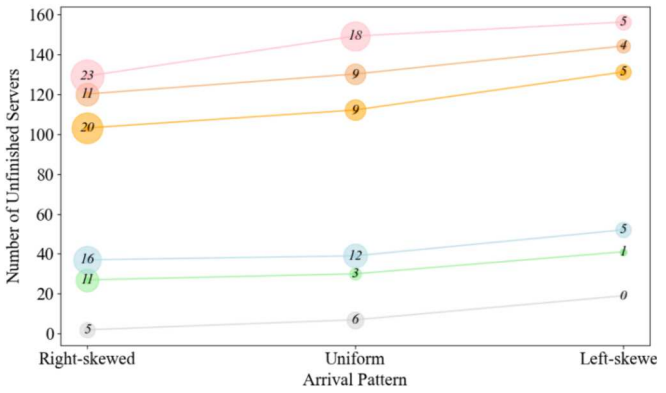


Fig. 13. Effect of arrival pattern on unfinished and tardy servers.

used the following formula:

New due date = Arrival time + Test processing time + Uniform (0, Max time of the planning horizon- Arrival time – Test processing time)

This formula calculates the new due date by adding the arrival time and test processing time for each server and then adding a random buffer time. This buffer time is drawn from a uniform distribution ranging from 0 to an upper limit calculated by subtracting the arrival and test processing times from the maximum time of the planning horizon.

Our findings reveal that tighter due dates result in an increase in the number of tardy servers in all experiments. This is intuitive because, while there may be capacity available in the future to finish the servers, there may not be enough capacity available now that could finish those servers prior to the due date. However, the quantity of unfinished servers remained unchanged, as depicted in Fig. 14.

We also examined the variation in the number of tardy servers across different lookahead window lengths, particularly under tighter due dates, within a system of fixed capacity and varying server sizes (300 and 400). Fig. 15(a) and (b) illustrate the impact of lookahead window lengths (1, 5, and 10 days) on the number of tardy servers (y-axis) for both 300 and 400 servers, across various fixed 2TC percentages. The x-axis displays combinations of these specific 2TC percentages with their respective lookahead window lengths.

Our findings indicate that increasing the lookahead window length from 1 to 5, and subsequently to 10 days, significantly reduces the number of tardy servers, irrespective of the server size within a system of fixed capacity. This reduction is attributed to the system's enhanced ability to access and utilize future demand and capacity information. This allows the optimization algorithm to allocate resources more efficiently, avoiding resource conflicts, which in turn leads to fewer tardy servers. The size of the servers, whether 300 or 400, has a negligible effect on tardiness due to the system reaching its capacity; beyond this threshold, additional servers do not significantly impact tardiness.

6. Summary and future research

Our research introduces an Online Dynamic Dual Bin Packing with Lookahead (OD-DBP-LA), a time-driven decision-making, approach that is based on the advanced certainty of future information for the computer server manufacturing industry, specifically focusing on the final test process. OD-DBP-LA aims to minimize the overall penalty incurred from not assigning servers (customer orders), while prioritizing the assignment of servers with earlier due dates. It addresses gaps in existing online scheduling literature by considering the effects of changes in the lookahead window on assignment and decision-making processes, while also incorporating several practical considerations not previously considered in online bin packing literature, such as server-test cell compatibility, test cell requirements, variable testing facility capacity, and due date prioritization to maximize the number of servers tested.

OD-DBP-LA generalizes the existing Online Bin Packing with

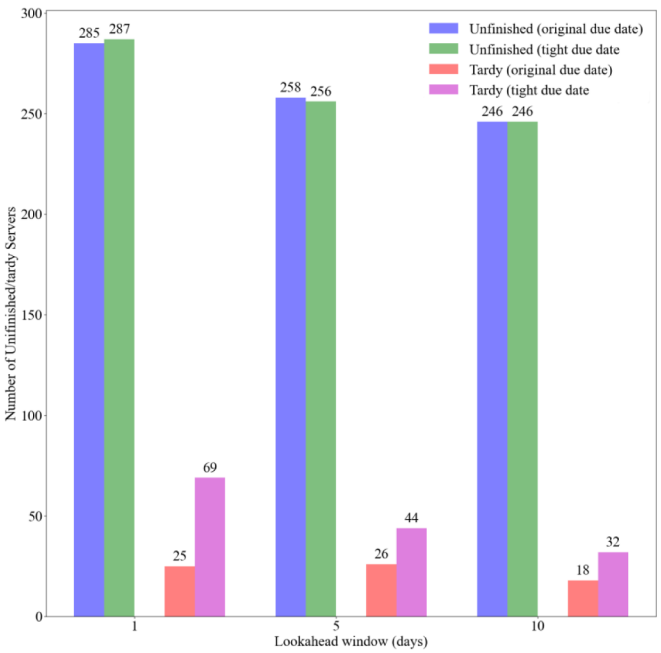
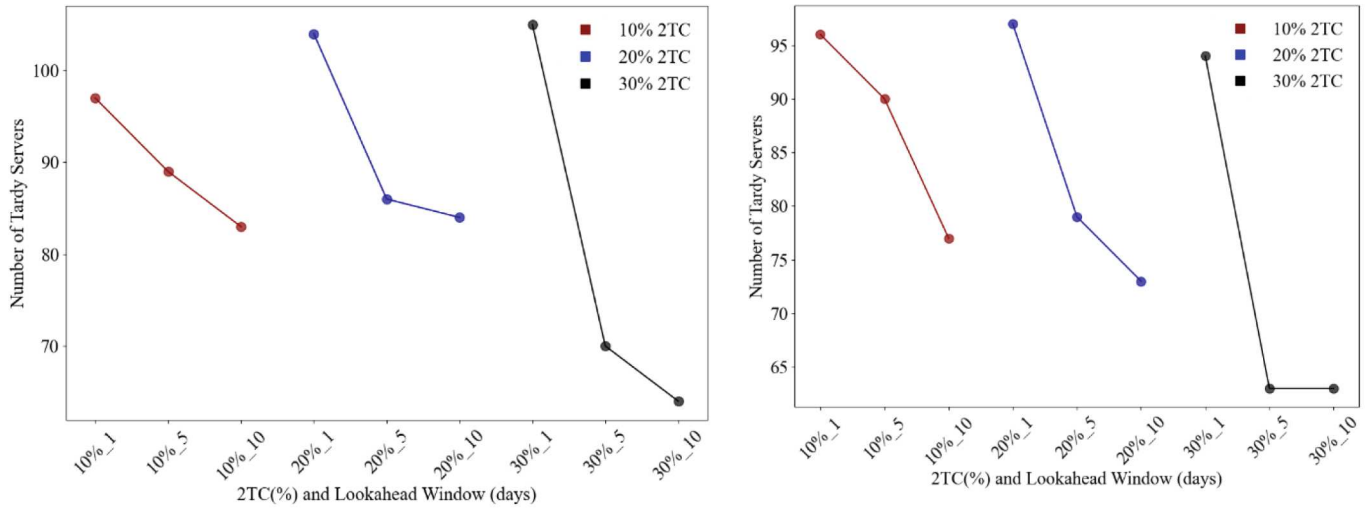


Fig. 14. Comparison of unfinished and tardy servers with original and tight due times across lookahead windows.

Lookahead (OBP-LA) in three directions: (i) through the time lookahead (LA) concept, where the lookahead is determined by the length of a time-window during which server arrival is known with certainty (compared to request lookahead), (ii) the dual bin packing (DBP) concept, and (iii) the dynamic concept (in OD). The inclusion of time lookahead, dual bin packing, and dynamic strategies ensures a faithful representation of the online decision-making process during the final testing phase of computer server manufacturing. In addition, the consideration of minimizing penalties for unassigned servers and prioritizing servers with earlier due dates in OD-DBP-LA allows decision-makers to effectively balance operational efficiency with timely fulfillment of customer demands.

We modeled OD-DBP-LA as an integer linear program and proposed a '2-phase' computational framework that integrated a metaheuristic based on GA framework with a mathematical programming approach to solve each deterministic, lookahead window-specific subproblem, addressing the challenges posed by two test cells requirement constraints and the problem's size. The set of testing facility we selected resembles that of our industry partner's site. Due to limited demand data available from our partner, we generated synthetic server arrival data, incorporating 'left-skewed,' 'right-skewed,' and 'uniform' distributions, and considered a time horizon of 13 weeks, corresponding to one quarter of a year. The key findings from our study are as follows:

- Longer lookahead windows lead to high quality results, but their effectiveness in improving outcomes diminishes with higher testing capacity and smaller server sizes.
- Server arrival patterns significantly impact outcomes during the final testing process. Right-skewed arrivals, where most servers come early in the planning horizon, allow such servers to have more time and available capacity for completion, resulting in fewer unfinished servers, but more tardy servers due to capacity overload and bottlenecks. Conversely, left-skewed arrivals, with most servers coming late in the horizon, lead to more unfinished servers due to capacity mismatches in a shorter time span, but fewer tardy servers as the limited number of processed servers are more likely to be completed on time.



(a) 300 servers (b) 400 servers

Fig. 15. Effect of numbers of servers on the tardy measure.

- If the lookahead window length is short, servers requiring short testing times are more likely to be unfinished, whereas with longer lookahead, servers needing longer testing times tend to remain unfinished. In a short window scenario, such as a 1-day lookahead, servers are selected randomly due to their indistinguishable nature in single-period assignments, but a 10-day lookahead allows the algorithm to distinguish processing times, thus prioritizing servers with shorter testing times to reduce the total number of unfinished servers. Moreover, servers that require 2TC are disproportionately unfinished compared to their representation in the original data.
- If the average lead time for servers decreases, the benefits of a longer lookahead become more pronounced, resulting in fewer tardy servers compared to using a shorter lookahead, without impacting the magnitude of unfinished servers.

These findings offer practical implications for decision-makers in server manufacturing. Server manufacturing decision-makers can use our approach to determine optimal decision-making frequency to improve the outcomes. Furthermore, decision-makers can weigh the trade-off between investing in information technology and engaging in client negotiations to increase server arrival certainty against the costs of expanding system capacity and accommodating smaller server sizes. Additionally, decision-makers can quantify the impact of server arrival patterns on server completion (on-time, tardy, or unfinished), enabling them to better design the system to handle demand variability and improve capacity utilization. Furthermore, server manufacturing and testing operations should adjust their lookahead strategies based on server testing times and specific needs their servers and the constraints of their testing facilities. For short lookahead windows, strategies need to be developed to minimize the risk of leaving servers with shorter testing times unfinished due to the randomness of server selection. Conversely, with longer lookahead windows, the focus should shift towards effectively managing and scheduling servers requiring longer testing times to prevent them from remaining unfinished. Additionally, special attention should be given to servers requiring two test cells (2TC), possibly through dedicated resources or adjusted scheduling priorities, to avoid disproportionately high unfinished rates. Finally, our findings on tighter due dates demonstrate how the system behaves under these conditions, highlighting the need for decision-makers to negotiate more realistic deadlines with clients. By understanding the effects of tight due dates, decision-makers can better plan and allocate resources to ensure timely delivery, thus avoiding both unfinished and tardy orders.

The insights from our model extend beyond computer server manufacturing and can be applied to various manufacturing systems that require real-time and dynamic task assignment to fixed resources. Manufacturing environments with high variability in demand and complex scheduling requirements can benefit by identifying the optimal length of lookahead windows, as the effectiveness of these windows diminishes with higher capacity and smaller task sizes. Additionally, the impact of different arrival patterns on task completion rates is relevant to industries where timing and order of arrivals significantly affect operational efficiency. By applying these insights, manufacturers across diverse sectors — automotive (e.g., Ford, GM, Toyota), aerospace, consumer electronics (e.g., Apple, Samsung, Dell), packaging and logistics, healthcare, and capital machinery (e.g., GE, Johnson & Johnson)—can enhance their resource allocation and scheduling strategies, ensuring better capacity management and improved outcomes in terms of timely task completion. This adaptability demonstrates the model's applicability to a wide range of manufacturing settings, enabling them to handle varying demand scenarios more efficiently.

Future research in this domain could explore the use of sliding windows instead of static ones to more effectively manage uncertainties in the testing process. Considering that the accuracy of information within a lookahead window may vary in real-world scenarios, data analytics could be employed to enhance forecasts of future server arrivals, improving decision-making. Moreover, incorporating uncertainty factors such as server arrival times, testing durations, machine failures, and server repairs into the testing strategy could optimize a wider range of performance metrics. A focus on minimizing energy consumption and carbon dioxide emissions would not only offer economic advantages but also promote environmental sustainability in manufacturing enterprises within both electronics and non-electronics sectors. Further, exploring different matheuristic or pure metaheuristic algorithms such as Ant Colony Optimization, Differential Evolution, or Simulated Annealing to potentially enhance solution quality and computational efficiency seems viable. This approach underlines the importance of sustainable practices in increasing operational efficiency and reducing the environmental impact of manufacturing processes.

CRediT authorship contribution statement

Mahmud Parvez: Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation. **Pratik J. Parikh:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology,

Investigation, Funding acquisition, Formal analysis, Conceptualization. **Faisal Aqlan:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Data curation, Conceptualization. **Md. Noor-E-Alam:** Writing – review & editing, Validation, Supervision, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgment

This research was partially supported by a grant from the National Science Foundation (CMMI #2038325).

Appendix A Additional details on the 2-phase computational framework

Function GA($w, m, S_m, E_m, T_m, K, P_k, C_k, W_k, H_k, L_k, D_k, R_z, \boldsymbol{A_{kt}}, \boldsymbol{W_{ij}}, \boldsymbol{H_{ij}}, \boldsymbol{L_{ij}}$):
  Population = Generate initial population
  while *stopping criteria not met* **do:**
    EvaluatePopulation(Population, $K, P_k, C_k, W_k, H_k, L_k, D_k, R_z, \boldsymbol{A_{kt}}, \boldsymbol{W_{ij}}, \boldsymbol{H_{ij}}, \boldsymbol{L_{ij}}$)
    Sort the individuals from the Population based on fitness
    Select parents for recombination
    Create offspring from parents
    Mutate the offspring
    Set new Population
  EndWhile
  Convert the final solution and return the data
End Function
Function MathModel($w, m, S_m, E_m, T_m, K, P_k, C_k, W_k, H_k, L_k, D_k, R_z, \boldsymbol{A_{kt}}, \boldsymbol{W_{ij}}, \boldsymbol{H_{ij}}, \boldsymbol{L_{ij}}$, Solution_Phase1):
  Define the required variables, parameters, objective function, and constraints in DOcplex
  Extract 2TC server assignments from Solution_Phase1 to appropriately fix the assignments
  Assign single cell requiring servers using DOcplex
  Return the solution obtained from DOcplex
End Function
Function EvaluatePopulation (Population, $K, C_k, P_k, W_k, H_k, L_k, D_k, R_z, \boldsymbol{A_{kt}}, \boldsymbol{W_{ij}}, \boldsymbol{H_{ij}}, \boldsymbol{L_{ij}}$):
  CompatibleSolutions = CompatibilityChecking(Population, $P_k, W_k, H_k, L_k, D_k, R_z, \boldsymbol{A_{kt}}, \boldsymbol{W_{ij}}, \boldsymbol{H_{ij}}, \boldsymbol{L_{ij}}$)
  CostObjective = CostCalculation(CompatibleSolutions,K, C_k)
  Return CostObjective
End Function
Function CompatibilityChecking(Population, $P_k, W_k, H_k, L_k, D_k, R_z, \boldsymbol{A_{kt}}, \boldsymbol{W_{ij}}, \boldsymbol{H_{ij}}, \boldsymbol{L_{ij}}$):
  Length = Calculate the total number of chromosomes in the Population
  For $i = 0$ *to Length* **do:**
    solution = Population[i]
    Set the solution as compatible (feasible)
    While *all servers in solution are not checked* **do**:
      server = Select a server
      Cell1 = Extract the cell for the selected *server*
      If ($D_{k=server} == 1$):
        Cell2 = Extract adjacent cell of Cell1
      If (*server is not compatible with Cell*1 *(and Cell*2 *if,* $D_{k=server} == 1$):
        Set the solution as incompatible (infeasible)
        Break
      Continue to the next *server*
    EndWhile
  EndFor
  Return the compatible solutions
End Function
Function CostCalculation (CompatibleSolutions, K, C_k):
  LengthOfServers = Calculate the total number of servers in K
  LengthOfSolutions = Calculate the total number of solutions in CompatibleSolutions
  CostSet = initialize a set to return the objective value of each solution
  For $i = 0$ to LengthOfSolutions **do**:
    CostOfSolution = 0
    Solution = CompatibleSolutions[i]
    For $j = 0$ *to LengthOfServers* **do:**
      SERVER = $K[j]$
      IF (*SERVER does not exist in Solution*):
        CostOfSolution += $C_{k=SERVER}$
    EndFor
    Add CostOfSolution to CostSet
  EndFor
  Return CostSet
End Function

Pseudocode (GA and MathModel functions).

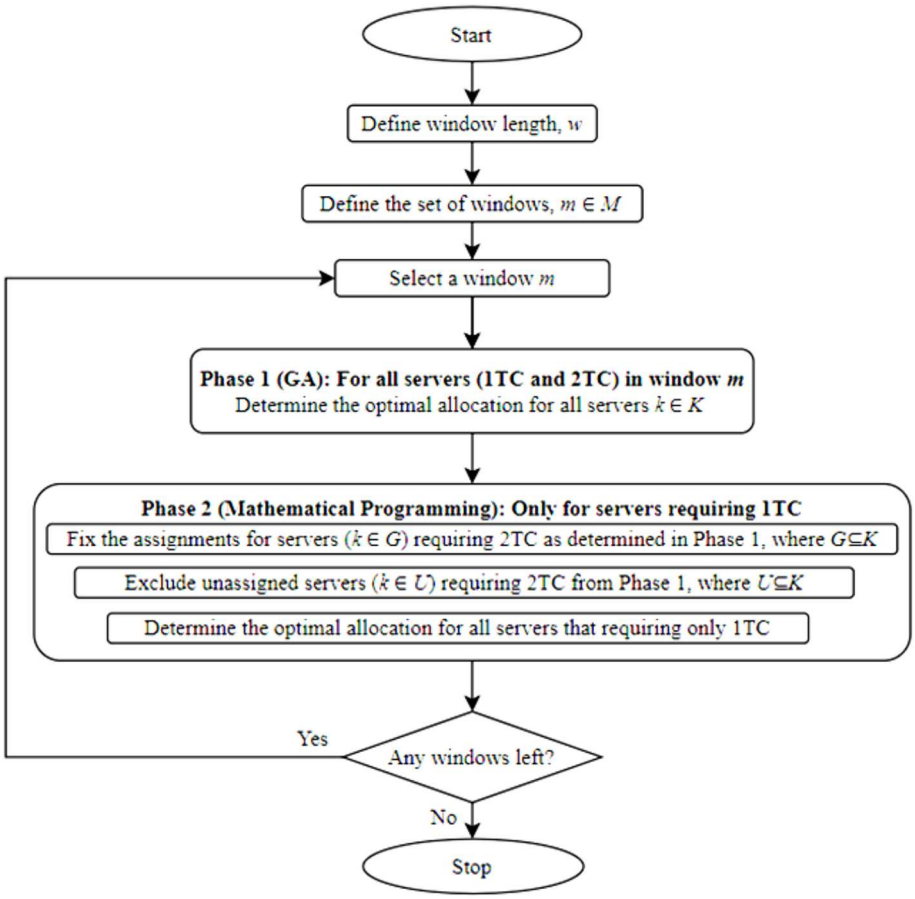


Figure 16. Flowchart of the '2-phase' computational framework.

References

Angelopoulos, S., Dürr, C., Kamali, S., Renault, M. P., & Rosén, A. (2018). Online bin packing with advice of small size. *Theory of Computing Systems, 62*, 2006–2034.

Angelopoulos, S., Kamali, S., & Shadkami, K. (2023). Online bin packing with predictions. *Journal of Artificial Intelligence Research, 78*, 1111–1141.

Aqlan, F., Lam, S. S., & Ramakrishnan, S. (2014). An integrated simulation-optimization study for consolidating production lines in a configure-to-order environment. *International Journal of Production Economics, 148*, 51–61.

Assmann, S., Johnson, D., Kleitman, D., & Leung, J. (1984). On a dual version of the one-dimensional bin packing problem. *Journal of Algorithms, 5*, 502–525.

Balseiro, S., Kroer, C., & Kumar, R. (2023). Online resource allocation under horizon uncertainty. In Abstract Proceedings of the 2023 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (pp. 63–64).

Berndt, S., Jansen, K., & Klein, K. M. (2020). Fully dynamic bin packing revisited. *Mathematical Programming, 179*(1–2), 109–155.

Bódis, A., & Balogh, J. (2019). Bin packing problem with scenarios. *Central European Journal of Operations Research, 27*, 377–395.

Böhm, M., Sgall, J., van Stee, R., & Veselý, P. (2017). Online bin stretching with three bins. *Journal of Scheduling, 20*, 601–621.

Boyar, J., Favrholdt, L. M., Kamali, S., & Larsen, K. S. (2021). Online bin covering with advice. *Algorithmica, 83*, 795–821.

Boyar, J., Favrholdt, L. M., Larsen, K. S., & Nielsen, M. N. (2001a). The competitive ratio for on-line dual bin packing with restricted input sequences. *Nordic Journal of Computing, 8*(4), 463–472.

Boyar, J., Kamali, S., Larsen, K. S., & López-Ortiz, A. (2016). Online bin packing with advice. *Algorithmica, 74*, 507–527.

Boyar, J., Larsen, K. S., & Nielsen, M. N. (2001b). The accommodating function: A generalization of the competitive ratio. *SIAM Journal on Computing, 31*(1), 233–258.

Buchbinder, N., Fairstein, Y., Mellou, K., Menache, I., & Naor, J. (2021). Online virtual machine allocation with lifetime and load predictions. *ACM SIGMETRICS Performance Evaluation Review, 49*(1), 9–10.

Bukkur, K. M. M. A., Shukri, M. I., & Elmardi, O. M. (2018). A review for dynamic scheduling in manufacturing. *The Global Journal of Researches Engineering, 18*(5-J), 25–37.

Burcea, M. (2014). Online dynamic bin packing (Doctoral dissertation). University of Liverpool.

Cheng, Y., Tao, F., Liu, Y., Zhao, D., Zhang, L., & Xu, L. (2013). Energy-aware resource service scheduling based on utility evaluation in cloud manufacturing system. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 227*(12), 1901–1915.

Coffman, E. G., Jr., Garey, M. R., & Johnson, D. S. (1983). Dynamic bin packing. *SIAM Journal on Computing, 12*(2), 227–258.

Dunke, F., & Nickel, S. (2016). A general modeling approach to online optimization with lookahead. *Omega, 63*, 134–153.

Dunke, F., & Nickel, S. (2021). Exact distributional analysis of online algorithms with lookahead. *4OR, 19*(2), 199–233.

Epstein, L., & Favrholdt, L. M. (2003). On-line maximizing the number of items packed in variable-sized bins. *Acta Cybernetica, 16*(1), 57–66.

Gupta, B. D., & Palis, M. A. (2001). Online real-time preemptive scheduling of jobs with deadlines on multiple machines. *Journal of Scheduling, 4*(6), 297–312.

Gupta, V., & Radovanović, A. (2020). Interior-point-based online stochastic bin packing. *Operations Research, 68*(5), 1474–1492.

Guruganesh, G. (2018). Topics in Approximation and Online Algorithms (Doctoral dissertation). Carnegie Mellon University Pittsburgh, PA.

Herrmann, J. W. (Ed.). (2006). *Handbook of production scheduling* (Vol. 89). Springer Science & Business Media.

Jaillet, P., & Lu, X. (2011). Online resource allocation problems. *Rock & Soil Mechanics, 86*, 3701–3704.

Kocsi, B., Matonya, M. M., Pusztai, L. P., & Budai, I. (2020). Real-time decision-support system for high-mix low-volume production scheduling in industry 4.0. *Processes, 8*(8), 912.

Kurilova-Palisaitiene, J., Sundin, E., & Poksinska, B. (2018). Remanufacturing challenges and possible lean improvements. *Journal of Cleaner Production, 172*, 3225–3236.

Li, Y., Tang, X., & Cai, W. (2014). On dynamic bin packing for resource allocation in the cloud. In Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (pp. 2–11).

Lu, X. (2013). Online optimization problems (Doctoral dissertation). Massachusetts Institute of Technology.

Mezmaz, M., Melab, N., Kessaci, Y., Lee, Y. C., Talbi, E.-G., Zomaya, A. Y., & Tuyttens, D. (2011). A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for

cloud computing systems. *Journal of Parallel and Distributed Computing, 71*(11), 1497–1508.

Murhekar, A., Arbour, D., Mai, T., & Rao, A. B. (2023). Brief announcement: Dynamic vector bin packing for online resource allocation in the cloud. In Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures (pp. 307-310).

Nasr, N., Haselkorn, M., Parnel, K., Burn, V., & Hanson, F. (2017). *Technology roadmap for remanufacturing in the circular economy*. Rocherster, NY: Golisano Institute for Sustainability, Rochester Institute of Technology.

Ojha, A., Agarwal, M., Singhal, A., Sarkar, C., Ghosh, S., & Sinha, R. (2021). A generalized algorithm and framework for online 3-dimensional bin packing in an automated sorting center. In *2021 Seventh Indian Control Conference (ICC)* (pp. 135–140). IEEE.

Parikh, P. J., & Meller, R. D. (2008). Selecting between batch and zone order picking strategies in a distribution center. *Transportation Research Part E: Logistics and Transportation Review, 44*(5), 696–719.

Peeters, M., & Degraeve, Z. (2006). Branch-and-price algorithms for the dual bin packing and maximum cardinality bin packing problem. *European journal of operational research, 170*(2), 416–439.

Pinedo, M. L. (2012). *Scheduling* (Vol. 29). New York: Springer.

Polyakovskiy, S., & M'Hallah, R. (2018). A hybrid feasibility constraints-guided search to the two-dimensional bin packing problem with due dates. *European journal of operational research, 266*(3), 819–839.

Renault, M. P., Rosén, A., & van Stee, R. (2015). Online algorithms with advice for bin packing and scheduling problems. *Theoretical Computer Science, 600*, 155–170.

Saha, C. (2015). *A framework for managing uncertainty using decision support system in a closed-loop supply chain business environment (Ph.D. dissertation)*. Binghamton, NY: Binghamton University.

Seiden, S. S. (2002). On the online bin packing problem. *Journal of the ACM (JACM), 49*(5), 640–671.

Vijayakumar, B., Parikh, P. J., Scott, R., Barnes, A., & Gallimore, J. (2013). A dual bin-packing approach to scheduling surgical cases at a publicly-funded hospital. *European Journal of Operational Research, 224*(3), 583–591.

Wang, X., Wang, S., & Xu, Q. (2022). Simultaneous production and maintenance scheduling for refinery front-end process with considerations of risk management and resource availability. *Industrial & Engineering Chemistry Research, 61*(5), 2152–2166.

Zhang, Y., Wang, J., Liu, S., & Qian, C. (2017). Game theory based real-time shop floor scheduling strategy and method for cloud manufacturing. *International Journal of Intelligent Systems, 32*(4), 437–463.

Zhao, H., She, Q., Zhu, C., Yang, Y., & Xu, K. (2021). Online 3D bin packing with constrained deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 1, pp. 741-749).

Zhao, H., Zhu, C., Xu, X., Huang, H., & Xu, K. (2022). Learning practically feasible policies for online 3D bin packing. *Science China Information Sciences, 65*(1), Article 112105.