International Journal of Production Research

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tprs20

Predictive vehicle dispatching method for overhead hoist transport systems in semiconductor fabs

Jiansong Wan & Hayong Shin

To cite this article: Jiansong Wan & Hayong Shin (2021): Predictive vehicle dispatching method for overhead hoist transport systems in semiconductor fabs, International Journal of Production Research, DOI: 10.1080/00207543.2021.1910870

To link to this article: https://doi.org/10.1080/00207543.2021.1910870

Published online: 13 Apr 2021.

INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH
https://doi.org/10.1080/00207543.2021.1910870

Taylor & Francis
Taylor & Francis Group

Predictive vehicle dispatching method for overhead hoist transport systems in semiconductor fabs

Jiansong Wan and Hayong Shin

Department of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology, Daejeon, South Korea

ABSTRACT

We propose to use information regarding the fab's future state for Overhead Hoist Transport (OHT) dispatching, which is named as 'predictive dispatching' in this paper. Unlike conventional dispatching methods, two kinds of information are additionally considered in our proposed methods: the expected arrival time of jobs in the near future and the time needed for occupied vehicles to become idle. We firstly develop Basic Predictive Dispatching (BPD) under the assumption that job arrival time prediction is error-free. We demonstrate that BPD consistently surpasses conventional benchmark dispatching methods, even when job arrival time prediction contains a certain level of error. However, as the level of error increases, the performance of BPD deteriorates. To improve BPD's performance in the environment with prediction error, we take the certainty level of job arrival time prediction into consideration in our second method called Certainty Weighted Predictive Dispatching (CWPD). Both BPD and CWPD formulate the OHT dispatching problem as a linear assignment problem, but two different matching cost functions are employed separately. By conducting experiments on a sample semiconductor fab, we validate the effectiveness of our proposed approaches. The superiority of CWPD over BPD in the environment with prediction error is also verified.

ARTICLE HISTORY
Received 9 September 2020
Accepted 18 March 2021

KEYWORDS
Predictive control; vehicle dispatching; assignment problem; overhead hoist transport; semiconductor fabrication

1. Introduction

Semiconductor manufacturing is known as one of the most capital-intensive and technology-intensive industries in the world. Typical wafer fabrication is highly complicated and involves hundreds of repetitive processing steps, which requires wafers to be frequently transferred among different process machines (a.k.a. tools). Wafer transport was once performed by human operators. As the wafer size grows to 300 mm and then to 450 mm in recent years, Automatic Material Handling Systems (AMHS) have been widely adopted, forming the transportation backbone of semiconductor fabs. The goal of an AMHS is to reduce manufacturing cycle time and enhance tool utilisation (Sarin, Varadarajan, and Wang 2011).

When wafers are transferred between tools, they are carried in the Front Opening Unified Pod (FOUP) as a lot, which typically stores 25 wafers (Hwang and Jang 2020). OHT is generally recognised as the main transport instrument for the AMHS of wafer fabs. An OHT system consists of multiple automatic vehicles and guided overhead rails. OHTs perform wafer retrieval, wafer delivery, and parking tasks on the guided rails by following the instructions from the OHT Control Systems (OCS). Inefficient operation of OHTs always leads to undesirable delays of wafer transfer hence disrupting the production schedules of tools. Therefore, making the OHT system and tools operate harmoniously has become a significant challenge for semiconductor manufacturers. Particularly the dispatching and routing problems related to the OHT system have attracted considerable attention from academia (e.g. Schmaler et al. 2017; Lee, Lee, and Na 2018; Ahn and Park 2021).

This paper focuses on the dispatching of OHTs – the procedure of determining the assignment between wafer transfer requests and OHTs. The dispatching decision is usually made when a transfer request is released or when an occupied OHT finishes its assigned delivery request. It is a very challenging task due to the variability of wafer process times, the traffic of OHT vehicles, and the massive number of transportation requests (Aresi et al. 2019). Meanwhile, the quality of OHT dispatching decision directly affects AMHS's efficiency and then crucially influences the fab's overall performance. Therefore, a large number of studies have been done on developing effective OHT dispatching methods (e.g.

CONTACT Hayong Shin hyshin@kaist.ac.kr Department of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology, 291 Daehak-ro, Daejeon 34141, South Korea

Wang and Chen 2012; Qin, Zhang, and Sun 2013; Hu et al. 2020).

However, most of the previous works on OHT dispatching have merely paid attention to the fab's state at the dispatching moment. There is indeed some information regarding the fab's future state, e.g. the estimated arrival time of near-future transfer requests and the time needed for occupied OHTs to be idle, which is beneficial to make OHT dispatching decisions. Those kinds of information can be provided by the various modernised informative systems of a fab, e.g. the Manufacturing Execution System (MES). Moreover, such information is supposed to be more accessible and accurate with the advance of Industrial IoT technologies. Nevertheless, even so, few dispatching methods have considered such information. This study is to bridge the gap.

In this paper, we propose two novel predictive OHT dispatching methods called Basic Predictive Dispatching (BPD) and Certainty Weighted Predictive Dispatching (CWPD). In the two proposed dispatching methods, two kinds of information: (1) *the arrival time of near-future lot transfer requests* and (2) *the time needed for occupied vehicles to be idle*, are additionally considered to make a look-ahead OHT dispatching decision. Both BPD and CWPD formulate the OHT dispatching problem as a linear assignment problem, while two different matching cost functions are separately defined. BPD is proposed for the environment without prediction error in which the prediction on the arrival time of near-future transfer requests is error-free. In reality, however, the arrival time of future jobs usually can not be precisely estimated. Thus we further propose the improved version of BPD called CWPD to yield higher performance in the environment where the arrival time of near-future transfer requests is erroneous. The effectiveness of our proposed methods is verified by performing experiments in a sample fab under different assumptive environments.

The rest of the paper is structured as follows. In Section 2, a brief literature review on vehicle dispatching is addressed. The details of the problem we address and assumptions are presented in Section 3. Section 4 presents the proposed rules. Experiment results and analysis are described in Section 5. Finally, in Section 6, we draw conclusions.

2. Literature review

This section gives a brief review of the previous studies on OHT dispatching and Automatic Guided Vehicle (AGV) dispatching. These works generally can be classified into two categories: static vehicle dispatching rules and dynamic vehicle dispatching rules. The difference lies in whether dispatching decisions can be changed adaptively. The previous works related to predictive vehicle management are reviewed in a separate subsection. Hereafter, a vehicle refers to the vehicle unit in an OHT system or an AGV system; a job refers to the lot transfer request or lot.

2.1. Static vehicle dispatching

In static vehicle dispatching rules, the job-vehicle assignment decisions are not allowed to change once they are made. The existing static dispatching rules are mainly greedy heuristic rules. A few works have studied the mixed dispatching policy that integrates the single classic rules. Hu et al. (2020) has proposed a deep reinforcement learning based dispatching policy that combines five conventional dispatching rules: First Come First Served (FCFS), Shortest Travel Distance First (STDF) or Nearest Job First (NJF), Earliest Due Date (EDD), Longest Waiting Time (LWT), and Nearest Vehicle First (NVF). Their model is trained to select the appropriate dispatching rules and vehicles according to different situations. Kuo and Huang (2006) have proposed a multimission-oriented vehicle dispatching policy in which four dispatching rules are dynamically switched based on a fuzzy logic mechanism. By combining the advantage of the NJF and the LWT rule, Liao and Fu (2002) proposed the Modified Nearest Job First (MNJF) rule. When a vehicle becomes idle, if there exist jobs whose waiting time exceeds the predefined time factor, then the job with the longest waiting time will be selected by applying LWT. Otherwise, the NJF rule is executed. Factorised Nearest Job First (FNJF) is also a composite dispatching rule of the NJF and LWT. It selects a job for transport based on a cost function defined by the waiting time as well as the distance between the job and the available vehicles. Table 1 gives an overview of reviewed static dispatching methods.

2.2. Dynamic vehicle dispatching

Unlike static dispatching rules, dynamic dispatching rules allow changing job-vehicle assignment decisions adaptively to capture the constant change of fab situation. Liao and Wang (2006) proposed an effective dynamic dispatching policy called Differentiated Preemptive Dispatching Policy (DPD) to provide prioritised transport services for jobs with high priority. When an empty vehicle is reserved for a high-priority job, if another vehicle becomes empty and is closer to the initially reserved vehicle, the new idle vehicle will be assigned to the high-priority job, which changes the original assignment. Wang and Chen (2012) proposed an improved version of DPD called Heuristic Preemptive Dispatching policy

Table 1. Overview of static dispatching methods.

Dispatching rules	Types	Description
HP	Single	Job with highest priority is delivered first
FCFS	Single	Job with earliest arrival time is delivered first
STDF/NJF	Single	Job with shortest travel distance is moved first
EDD	Single	Job with earliest due date is selected first
LWT	Single	Job with the longest waiting time is selected first
NVF	Single	Vehicle with shortest distance is selected first
DQN-based (Hu et al. 2020)	Multiple	FCFS, STDF, EDD, LWT, and NVF are adaptively selected by DQN
Multi-mission (Kuo and Huang 2006)	Multiple	HP, EDD,LWT, and NJF are adaptively selected by fuzzy logic
MNJF (Liao and Fu 2002)	Multiple	NJF or LWT is selected by a predefined time factor
FNJF	Multiple	Job selected by a cost function of waiting time and distance

Table 2. Overview of dynamic dispatching methods.

Dispatching rules	Methodology	Description
DPD (Liao and Wang 2006)	Rule-based	Jobs with high priority enjoy preemptive transport service
HPD (Wang and Chen 2012)	Rule-based	Improved version of DPD
RBD (Kim et al. 2007)	Rule-based	Reassignment is made if there is better matching
HABOR (Kim et al. 2009)	Analytical	Vehicle dispatching is globally solved by Hungarian method
MHAFLC (Qin, Zhang, and Sun 2013)	Analytical	Hungarian method and fuzzy-logic-based weight adjusting

(HPD) that focuses on minimising transport blocking and waiting time for high priority jobs while creating a minimal negative impact on the transport for normal jobs. Kim et al. (2007) proposed an efficient vehicle reassignment logic called Reassignment Based Dispatching (RBD). When a delivery vehicle becomes empty, the new available vehicle will search the already assigned jobs but not loaded. If there exist such jobs that are closer to the new available vehicle than the initially reserved vehicle, the new available vehicle will select the best-fitted match. Afterward, they extended the idea of RBD and proposed the Hungarian Algorithm Based OHT Reassignment (HABOR) rule. In HABOR, the vehicle dispatching problem is formulated as an assignment problem and solved by the Hungarian algorithm (Kim et al. 2009). HABOR has been confirmed superior over RBD due to allowing multiple reassignments among the available vehicles and available jobs. Qin, Zhang, and Sun (2013) proposed a dynamic dispatching approach using a modified Hungarian algorithm and fuzzy-logic-based control (MHAFLC). In the proposed dispatching rule, job due date, job waiting time, and system load are thoroughly considered and used as the main elements of the cost function whose weight coefficients are adjusted by a fuzzy-logic-based control mechanism. Table 2 gives an overview of reviewed dynamic dispatching methods.

2.3. Look-ahead vehicle management

de Koster, Le-Anh, and van der Meer (2004) investigated the benefit of utilising the pre-arrival information of transfer requests for AGV dispatching and concluded that such information could reduce job-waiting times. However, they barely studied the case that the pre-arrival information of jobs is ideally accurate. They also raised an issue that looking ahead too far could increase the vehicle waiting time and harm the system performance. In our study, not only the benefit of accurate pre-arrival information, but the benefit of pre-arrival information with prediction errors is studied. Besides, we solved the issue they raised by giving a penalising cost on the assignment of vehicles to far-future jobs. Kim and Bae (2004) proposed a look-ahead dispatching algorithm for AGVs in an automated container terminal. But there is a huge difference between the AGV system of an automated container terminal and the OHT system of a wafer fab. Their main problem is more like a parallel machine scheduling problem with sequence-dependent setup times and precedence constraints among jobs. Huang and Lin (2016) has proposed a Pre-Dispatching Vehicle method (PDV) for the diffusion area in a 300 mm wafer fab. But the PDV focuses on pre-dispatching empty vehicles within a bay in which the complexity of dispatching decision making is largely reduced. Thus in PDV, the assignment between job and vehicle simply follows static heuristics rule, e.g. NVF. Gupta, Hasenbein, and Park (2020) proposed two look-ahead based dispatching policies, which used the information of estimated time that a busy vehicle will take to arrive at its destination given its current location. But they merely used future information related to the vehicle. Moreover, like previous works, only the greedy heuristic rule is used for the assignment between vehicles and jobs. Some studies have worked on the idle vehicle balancing by considering future information regarding vehicles and jobs for decision making (Chaabane et al. 2013; Schmaler et al. 2017). However, idle vehicle balancing is to flexibly determine the appropriate number of vehicles in different areas of fab according to job arrival dynamics, which has a fundamental difference with the dispatching problem addressed in this paper. Table 3 gives a comparison of reviewed literature with our work.

Table 3. Comparison of reviewed literature with our work.

Literatures	Research problem	Methodology	FVI	FJI
de Koster, Le-Anh, and van der Meer (2004)	Vehicle dispatching	Rule-based	×	✓
Kim and Bae (2004)	More like parallel machine scheduling	Analytical	✓	✓
Huang and Lin (2016)	Vehicle dispatching	Rule-based	×	✓
Gupta, Hasenbein, and Park (2020)	Vehicle dispatching	Rule-based	✓	×
Schmaler et al. (2017)	Idle vehicle balancing	Rule-based	✓	✓
Chaabane et al. (2013)	Idle vehicle balancing	Rule-based	✓	✓
Ours	Vehicle dispatching	Analytical	✓	✓

Note: FVI: future vehicle information. FJI: future job information.
✓: considered. ×: not considered.

3. Problem statement and assumptions

Figure 1 illustrates the general job delivery process in this study. Before arrival, the status of a job is future(F). When the job arrives at a specific bay, it becomes an unassigned waiting job(UW). It first enters the queue of the bay's loading point and then waits for retrieval of vehicles. The time interval between the job arrived moment and the vehicle assigned moment is defined as unassigned waiting time. As soon as a vehicle is assigned to the job, it starts retrieving, and its status changes from idle(I) to retrieval(R). The status of the job also changes into retrieval waiting(RW). The time that takes the vehicle to retrieve the job is termed as retrieval waiting time. After the vehicle reaches the loading point, the vehicle begins picking up the job, and its status changes to loading(L). The time spent on this procedure is called loading time. When loading is completed, the vehicle starts moving the job to its target bay. Delivery time is defined as the time spent on this procedure. After the vehicle arrives at the target bay's unloading point, the job is dropped off and immediately leaves the system. Time spent on dropping off the job is dubbed as unloading time. Both the loading time and unloading time are set to 10 s in the system. The total time spent on the whole delivery process is termed as lead time.

We take the arrival of a job as the dispatching decision point. That is to say: dispatching decisions are made every time a new job arrives. We define the look-ahead window as the time horizon that the pre-arrival information of jobs can be known. For instance, if the look-ahead window is set to 60 s, it suggests that the arrival information of jobs whose expected arrival time is within 60 s from the dispatching moment can be known in advance. Our objective is to minimise the average lead time of jobs. This objective can be restated as minimising the average waiting time (sum of unassigned waiting time and retrieval waiting time) of jobs since the dispatching method mainly influences the unassigned waiting time and retrieval waiting time.

Figure 2 presents a small size motivating example to state our research problem and goal more clearly. There are three vehicles: OHT1 just finished its previous job and now in idle status; OHT2 is retrieving job1; OHT3 is unloading a job and will become idle soon. There are three jobs: job1 is in being retrieved; job2 is waiting for assignment at the loading point of Tool1; job3 is arriving soon at the loading point of Tool2. We are now at dispatching decision making moment because of the arrival of Job2. In static dispatching scenarios, most traditional rules have merely considered the assignment between the unassigned waiting jobs(UW) and idle vehicles(I). But obviously, it is not a wise decision to match OHT1 with Job2 in this example because both OHT1 and OHT2 will travel a far distance for retrieving. Existing dynamic dispatching rules have made some extensions by allowing changing the job-vehicle assignment adaptively. They additionally consider retrieval waiting jobs(RW) as

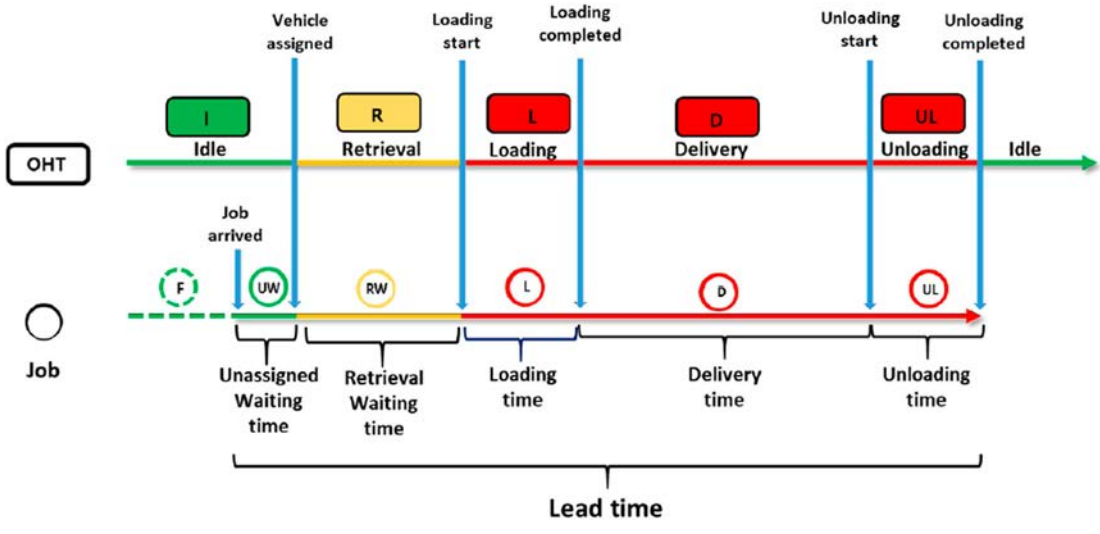


Figure 1. General job delivery process.

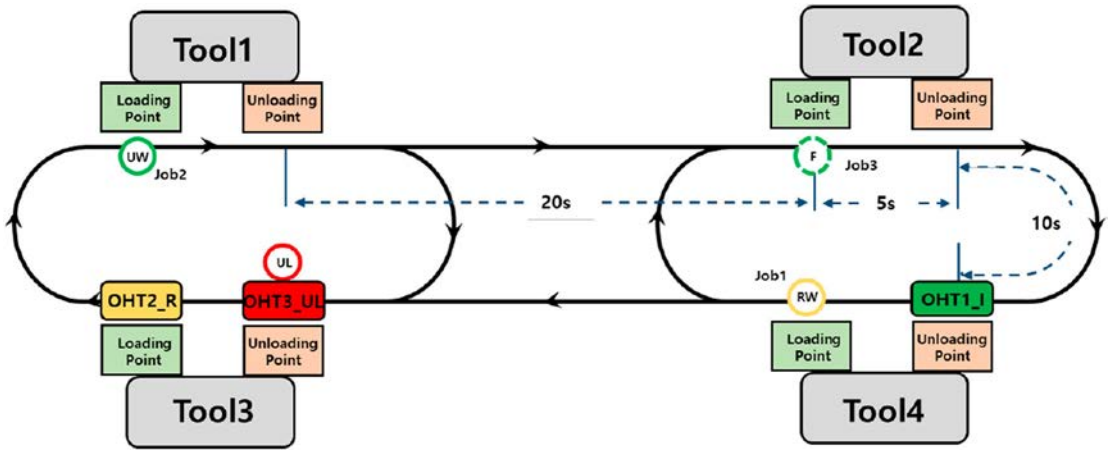


Figure 2. Motivating example.

dispatching available jobs and retrieval vehicles(R) as dispatching available vehicles. Thus dynamic dispatching rules can achieve a wiser decision: OHT1 matched with Job1 and OHT2 matched with Job2, which can apparently reduce the average waiting time of jobs. In our study, however, we take a more global look by additionally considering occupied vehicles and future jobs for decision making. Our study focuses on how to efficiently dispatch OHTs in a predictive way, e.g. OHT1 matched with Job1, OHT2 matched with Job2, OHT3 matched with Job3 in the motivating example.

4. Model description

In this section, we propose two dispatching methods: BPD and CWPD. In our model, the OHT dispatching problem is formulated as a linear assignment problem. Two matching cost functions are defined for the environment without prediction error and the environment with prediction error, respectively. Table 4 gives a summary of the notations that will typically be used throughout the following sections.

Table 4. Summary of notations.

Symbol	Explanation
J	available job set
V	available vehicle set
λ_{mn}	the arrival rate of transfer request from bay m to bay n
λ_m	the arrival rate of transfer request from bay m
EAT_j	the expected arrival time of job j
AAT_j	the actual arrival time of job j
t_{now}	the moment of dispatching
P_j	the probability of updating EAT_j for job j
TTA_j	time required for job j to arrive
$JT_v^{cl\rightarrow dest}$	Journey time from the current location of vehicle v to its destination
RLT_v	remaining loading time of vehicle v
$RULT_v$	remaining unloading time of vehicle v
RAT_j	relative arrival time of job j
TTC_v	time required for vehicle v to become idle
JWT_{jv}	job waiting time when job j is matched with vehicle v
c_{jv}	matching cost when job j is matched with vehicle v
cf_j	certainty factor of job j

4.1. Vehicle dispatching problem formulation

To make a dispatching decision, the first thing is to define the dispatching available jobs and dispatching available vehicles. In contrast to previous dispatching rules, we take the unassigned waiting jobs(UW), retrieval waiting jobs(RW), and future jobs(F) within the look-ahead window as dispatching available jobs. Besides idle vehicles and retrieval vehicles, occupied vehicles: loading vehicles(L), delivery vehicles(D), and unloading vehicles(UL) are also counted as dispatching available vehicles in our proposed methods. Even though the occupied vehicles can not perform other jobs right now, the time required for them to complete their assigned jobs can be approximately anticipated. We can determine the next jobs they should do as soon as they become unoccupied.

We formulate our vehicle dispatching problem as a linear assignment problem defined by equations (1)–(4). If the number of available jobs is larger than the number of vehicles, dummy vehicles will be added to make the same number of jobs and vice versa. The jobs assigned to dummy vehicles will temporarily remain unplanned. The vehicles matched with dummy jobs will keep their original status.

$$\text{Let} \quad x_{jv} = \begin{cases} 1 & \text{if job } j \text{ is assigned to vehicle } v \\ 0 & \text{if not} \end{cases}$$
$$c_{jv} = \text{cost of matching job } j \text{ with vehicle } v \tag{1}$$

$$\text{Minimize} \quad Z = \sum_{j}^{|J|} \sum_{v}^{|V|} c_{jv} x_{jv}$$

Subject to :

$$\sum_{v}^{|V|} x_{jv} = 1, \quad \text{for } j = 1, 2, \ldots, |J| \tag{2}$$

$$\sum_{j}^{|J|} x_{jv} = 1, \quad \text{for } v = 1, 2, \ldots, |V| \tag{3}$$

$$x_{jv} = \text{binary, for all } j \text{ and } v \tag{4}$$

To solve the linear assignment problem, the cost function of matching job j and vehicle v has to be well defined. Next, we elaborate on the two matching cost functions defined for the environment without prediction error and the environment with prediction error.

4.2. Matching cost definition

4.2.1. Basic predictive dispatching version

This cost function is defined for the environment where jobs' expected arrival time can be absolutely correct. Specifically, jobs' expected arrival time is exactly the same as their actual arrival time in the error-free environment. Therefore, all EAT_j in this subsection can be replaced by AAT_j.

For each job j in the available job set J, we define:

$$TTA_j = (EAT_j - t_{now})^+ \tag{5}$$

where TTA_j is the time required for job j to arrive at t_{now}. If job j has already arrived by t_{now}, then TTA_j is set to 0. In our motivating example, Job1 and Job2 have arrived, thus their TTA are 0. As for Job3, suppose $EAT_1 = 10$ s, $EAT_2 = t_{now} = 20$ s, $EAT_3 = 50$ s, thus $TTA_3 = 30$ s.

For each vehicle v in the available vehicle set V, we define:

$$TTC_v = \begin{cases} 0 & \text{if vehicle } v \text{ is in idle or retrieval status} \\ RLT_v + JT_v^{cl \to dest} + RULT_v & \text{if not} \end{cases} \tag{6}$$

where TTC_v is the time required for vehicle v to complete its currently assigned job. Since idle vehicles and retrieval vehicles can immediately go for a newly assigned job, if vehicle v is in idle or retrieval status, TTC_v is set to 0. The TTC_v of an occupied vehicle v is the sum of the remaining loading time, journey time from current location to destination, and the remaining unloading time. In our motivating example, OHT1 and OHT2 are unoccupied, thus their TTC are 0. But for OHT3, it has to first finish its current job before performing other jobs. Because OHT3 is in unloading status, thus $RLT_3 = 0$ and $JT_3^{cl \to dest} = 0$. Suppose the remaining unloading time is 5 s, then $TTC_3 = RLT_3 + JT_3^{cl \to dest} + RULT_3 = 5$ s.

In addition, RT_{jv} is defined as the retrieval time required when job j is assigned to vehicle v. To be more specific, it can be roughly considered as the journey time from the location where vehicle v is able to start retrieving earliest to the pick-up location of job j. For instance, if vehicle v is in idle or retrieval status, the location it can start retrieving earliest is its current location. But for the occupied vehicles, the locations they can start retrieving earliest should be their destinations because they can not retrieve other jobs until they complete their currently assigned jobs. In our motivating example, the locations to start retrieving earliest for three vehicles are all their current locations. Thus retrieval time here can be approximately calculated as the journey time from their current location to the pick-up location of their matching job. According to the travel time on the marked segment, we can obtain $RT_{11} = 5$ s, $RT_{12} = 55$ s, $RT_{13} = 60$ s, $RT_{21} = 40$ s, $RT_{22} = 10$ s, $RT_{23} = 15$ s, $RT_{31} = 15$ s, $RT_{32} = 35$ s, $RT_{33} = 40$ s.

By combining the three elements defined above, the preliminary cost function can be defined as:

$$c_{jv} = (TTC_v + RT_{jv} - TTA_j)^+ \tag{7}$$

The above cost function works well when the number of available jobs $|J|$ is smaller than the number of available vehicles $|V|$. Because in this condition, every job will indeed be assigned to a vehicle. Through the above cost function, the optimal matching for each job can be selected from the adequate vehicles so that the total dispatching cost will be minimised.

However, when the look-ahead window is large, $|J|$ will be larger $|V|$ and there will be some far-future jobs in the available job set.Their TTA_j may be even larger than $TTC_v + RT_{jv}$ for some vehicle v, which makes the matching cost becomes 0. Therefore, vehicles will be assigned to those far future jobs to minimise the total dispatching cost. This will cause vehicles to go to the origins of far-future jobs and wait for their arrival instead of performing the already arrived jobs and near-future jobs. This is exactly the issue raised by de Koster, Le-Anh, and van der Meer (2004). To resolve the issue, we can simply limit the maximum number of available jobs to the number of vehicles $|V|$. Next, we provide another solution for the issue, which can be considered as a general form of the preceding solution.

We add a new element to the cost function called relative arrival time (RAT) to resolve the issue. For each job j in the available job set J, we define:

$$RAT_j = EAT_j - \min\left\{EAT_1, EAT_2, \ldots, EAT_{|J|}\right\} \tag{8}$$

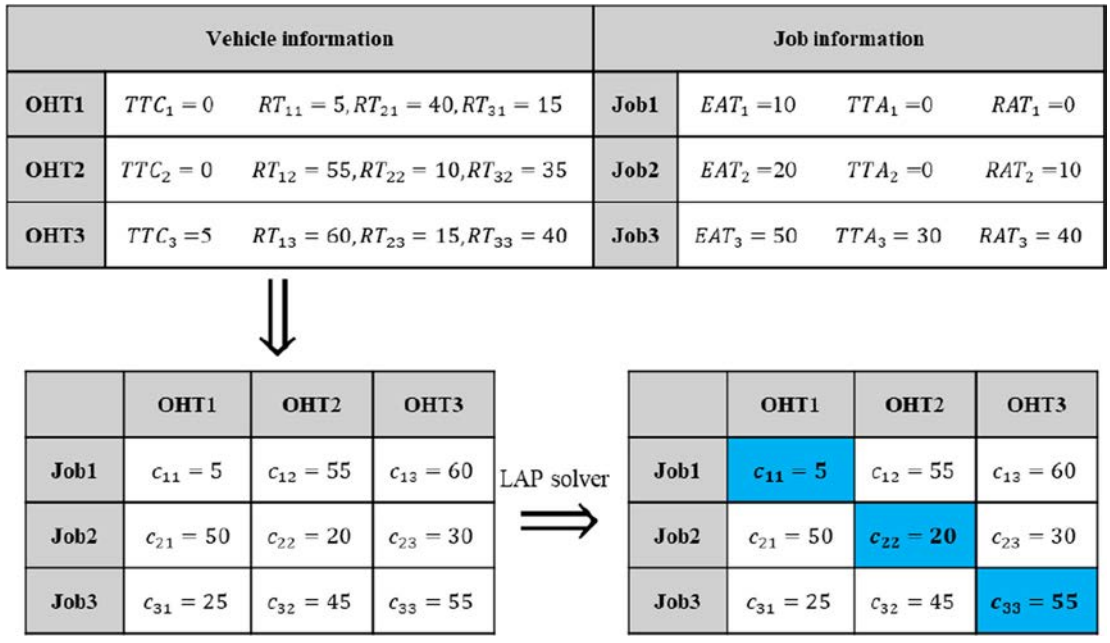


Figure 3. A manual solution to our motivating example by using the proposed BPD method.

To compactly represent the dispatching cost of matching job j and vehicle v, we define:

$$JWT_{jv} = (TTC_v + RT_{jv} - TTA_j)^+ \tag{9}$$

where JWT_{jv} is defined as the waiting time of job j when it is matched with vehicle v. It is the difference between the total time needed for vehicle v to retrieve job j and the time needed for job j to arrive. If $TTC_v + RT_{jv}$ is smaller than TTA_j, it means that vehicle v can arrive at the pickup location earlier than the arrival of job j, then $JWT_{jv} = 0$.

Finally, we propose the matching cost function, which is a combination of JWT_{jv} and RAT_j.

$$c_{jv} = JWT_{jv} + \alpha \cdot RAT_j \tag{10}$$

where α is the weighting parameter. Job j with a smaller RAT_j is supposed to have a relatively higher priority to be planned because the RAT_j of far-future jobs is large, thus will increase the matching cost. The value of α can be appropriately adjusted to different environmental settings. The preceding solution that limits the maximum number of available jobs to the number of vehicles $|V|$ is actually the case when α is dominantly large. The proof of the equivalence of the two solutions has been provided in Appendix 1.

To better explain the role of RAT_j, we make some changes on our motivating example. Suppose we only have OHT2, Job2 and Job3. Without RAT_j term, the cost of matching OHT2 with Job2 is $c_{22} = (TTC_2 + RT_{22} - TTA_2)^+ = 10$, the cost of matching OHT2 and Job3 is $c_{32} = (TTC_2 + RT_{32} - TTA_3)^+ = 5$. Because $c_{22} > c_{32}$, OHT2 will be matched with Job3 even though Job2 has already arrived and is waiting to be transported. If RAT_j term is introduced, we can first calculate $RAT_2 = 0, RAT_3 = 30$. Suppose $\alpha = 1$, then the cost of matching OHT2 and Job2 is $c_{22} = (TTC_2 + RT_{22} - TTA_2)^+ + RAT_2 = 10$, the cost of matching OHT2 and Job3 is $c_{23} = (TTC_2 + RT_{32} - TTA_3)^+ + RAT_3 = 35$. Now $c_{22} < c_{32}$, thus OHT2 will be matched with Job2, which is the desirable dispatching decision. The effectiveness of RAT_j can also be seen in Figure 8.

Figure 3 shows a manual solution to our motivating example by using the proposed BPD method. By integrating the vehicle information and job information, we get the matching cost matrix. The optimal matching solution can be obtained using any linear assignment problem solver. As we can see, the final result: OHT1 with Job1, OHT2 with Job2, and OHT3 with Job3, is actually the best decision we can expect in terms of our motivating example. OHT3 matched with Job3 means that as soon as OHT3 finishes unloading, it goes to the pick-up location of Job3. The waiting time of Job3 will be reduced.

4.2.2. Certainty weighted predictive dispatching version

As aforementioned, it is almost impossible to correctly estimate the arrival time of future jobs in reality. The prediction error may lead to severe performance degradation of BPD. To tackle this issue, we make a slight revision on BPD's matching cost function and name the proposed dispatching method using the revised matching cost function as CWPD.

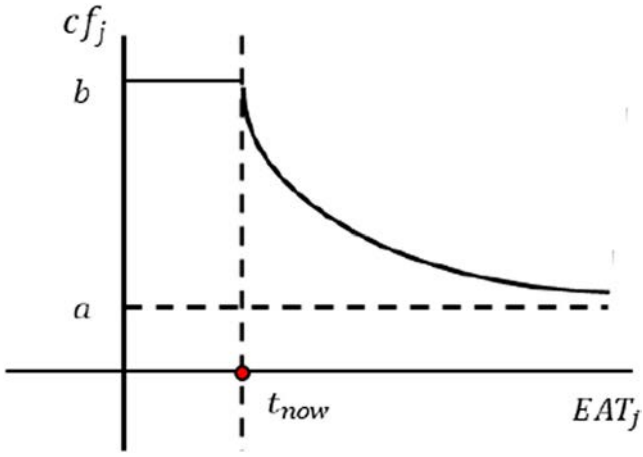


Figure 4. Relationship between cf_j and EAT_j.

The fundamental idea is that the jobs should be differentiated by the certainty of their EAT_j. Especially for the already arrived jobs, we are absolutely sure that their EAT_j is correct, which is their AAT_j. They should not be treated identically with future jobs as we have less certainty on the EAT_j of future jobs. Let cf_j be the certainty factor of job j with regard to EAT_j, and then it can be defined as:

$$cf_j = \begin{cases} b & \text{if } AAT_j \le t_{now} \\ \dfrac{b-a}{1+(EAT_j - t_{now})} + a & \text{if not} \end{cases} \tag{11}$$

where a and b are changeable constants that vary with different environments, $b > a$. The certainty factor of arrived jobs is set to b; the certainty factor of future jobs is inversely proportional to $EAT_j - t_{now}$ and in the range of $[a, b]$. It is commonly accepted that the near-future prediction is usually more accurate than the far-future prediction. By the same token, the prediction on the jobs whose EAT_j is closer to t_{now} should have higher certainty. Figure 4 displays the relationship between cf_j and EAT_j.

Then we propose the matching cost function for the environment with prediction error as:

$$c_{jv} = JWT_{jv} \times cf_j + \alpha \cdot RAT_j \tag{12}$$

By multiplying the certainty factor, the job with high certainty factors is more likely to be assigned with the vehicle that minimises its waiting time. The matching cost function defined in Equation (10) can be considered as a special case when b and a in Equation (11) are the same. A more detailed description on the role of cf_j is provided in Appendix 2.

5. Experiment and result analysis

In this section, we elaborate on the environment settings for experiments. A detailed analysis of the experiment results will also be presented. We first compare the performance of NVF, HABOR, and the proposed BPD method in the environment without prediction error. In the same environment, we further investigate the effect of the look-ahead window on the performance of BPD. The third and fourth experiments contrast the performance of BPD and CWPD in the environment with prediction error.

5.1. Illustration of sample semiconductor fab

The layout of the sample fab considered for this study is shown in Figure 5. It can be considered as the central loop of a whole unified fab which is comprised of 20 bays. Each

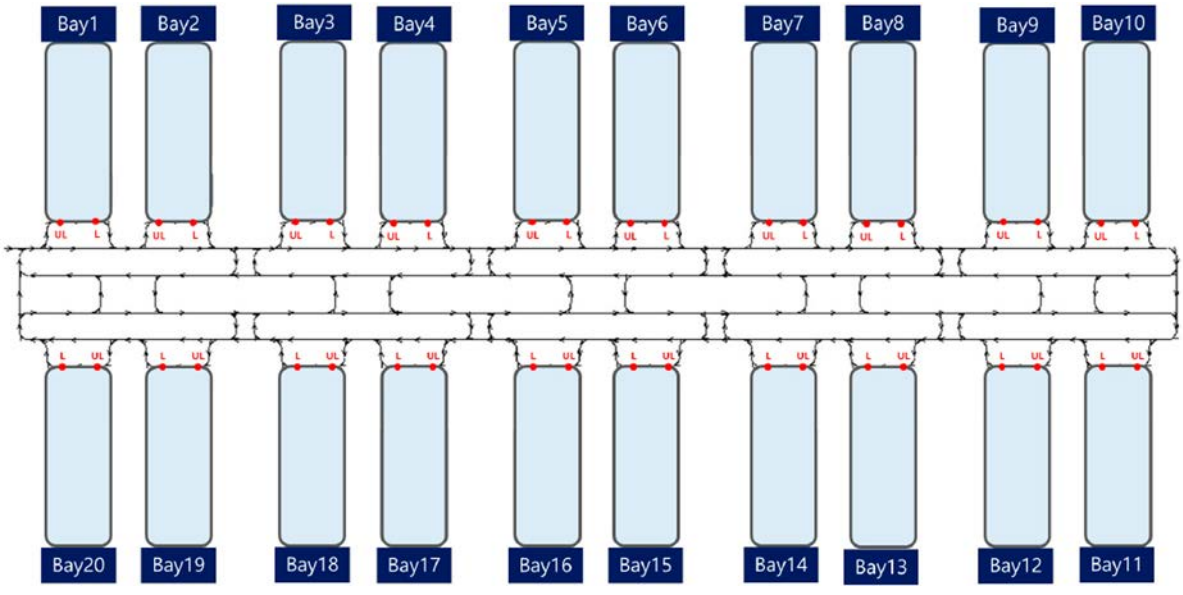


Figure 5. Track layout of sample fab.

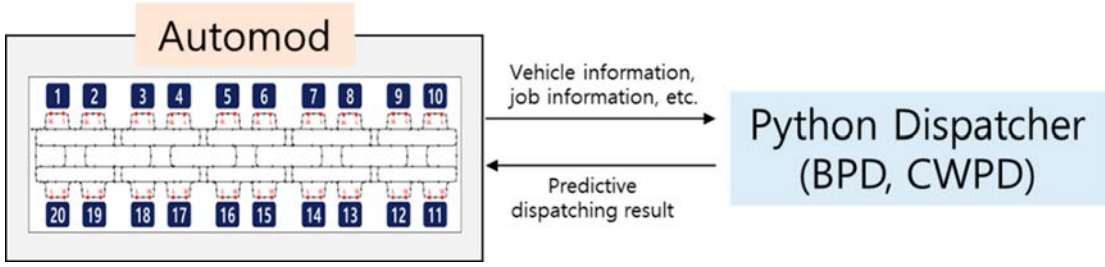


Figure 6. Implementation of the experiment environment.

bay has one loading point and one unloading point. The loading point and unloading point corresponds to the exit and entrance of a bay. The system has also been used as an experimental environment in the literature (Kim, Yu, and Jang 2016; Hwang and Jang 2020). The horizontal length and the vertical lengths are 128 m and 16 m, respectively. All the flow paths are unidirectional.

In the system, the details of the wafer processing are ignored. Transfer requests between the bays are directly generated to indicate that jobs are available to be performed. The arrivals of jobs between two different bays follow the independent Poisson Process. These settings have been adopted in previous literature (Kim et al. 2009; Bartlett et al. 2014; Hwang and Jang 2020). The expected interarrival time of jobs in the whole system is around 6 s. This makes the average utilisation of vehicles around 75% when applying the NVF dispatching rule, which is close to the real situation.

There are entirely 28 vehicles in the system. A vehicle can only perform one job at a time. The straight path and curve path velocity is 2 m/s and 0.5 m/s, respectively. Vehicles do not need to charge the battery, and no parking locations are designated for idle vehicles. When a delivery vehicle becomes idle, it remains at the destination location and waits for another call. If other vehicles want to pass the location, the idle vehicle will be pushed to the next bay's unloading point.

Figure 6 displays the structure of our experiment environment implementation. The sample fab is modelled using the most broadly used software in the semiconductor industry: *AutoMod*^{TM}. The proposed dispatching methods are implemented by Python. Socket communication is employed for information transmission between *AutoMod* and our dispatcher. Since we take the arrival of a job as the dispatching decision point, when a new job becomes available, the sample fab system sends the necessary information regarding the available jobs and the available vehicles to the Python dispatcher. The dispatcher formulates a linear assignment problem based on the received information and sends the optimal dispatching results back to the sample fab system. Vehicles are dispatched strictly according to the result obtained from the Python dispatcher.

We set a 24-simulation hour warm-up period for each environment in order that the system can reach the steady state. Then we collect data for 24 simulation hours after the warm-up period. In order to yield reliable simulation results, each simulation is repeated five times. In all the following experiments, we have limited the maximum number of available jobs to the number of vehicles. According to the discussion of Appendix 1, the RAT_j term becomes unnecessary in this condition. However, to show the completeness of our method, the weighting parameter α in the two matching cost functions is set as 1.

5.2. Environment without prediction error

In the error-free environment, we assume that the EAT_j of each job j is precisely the same as AAT_j. We denote the BPD method with 0 s and 20 s look-ahead window as BPD(0) and BPD(20), respectively. The first experiment compares the performance of NVF, HABOR, BPD(0), and BPD(20). The difference between the four dispatching methods is displayed in Table 5.

Figure 7 illustrates a performance comparison of the four dispatching methods. As expected, different dispatching methods have an insignificant influence on the average delivery time of jobs. However, we can apparently observe that BPD(0) and BPD(20) achieve better results than HABOR and NVF concerning the average lead time and the average waiting time. HABOR performs better than the NVF rule because it can capture the dynamic change of the fab state. BPD(0) shows higher efficiency than HABOR because it additionally considers occupied vehicles as dispatching available vehicles. The reason that BPD(20) performs better than BPD(0) can be attributed to utilising the pre-arrival information of

Table 5. Summary of NVF, HABOR, BPD(0) and BPD(20).

Dispatching methods	Property	Types of available jobs	Types of available vehicles
NVF	Static	UW	I
HABOR	Dynamic	UW, RW	I, R
BPD(0)	Dynamic	UW,RW	I, R, **L**, **D**, **UL**
BPD(20)	Dynamic	UW,RW, **F**	I, R, **L**, **D**, **UL**

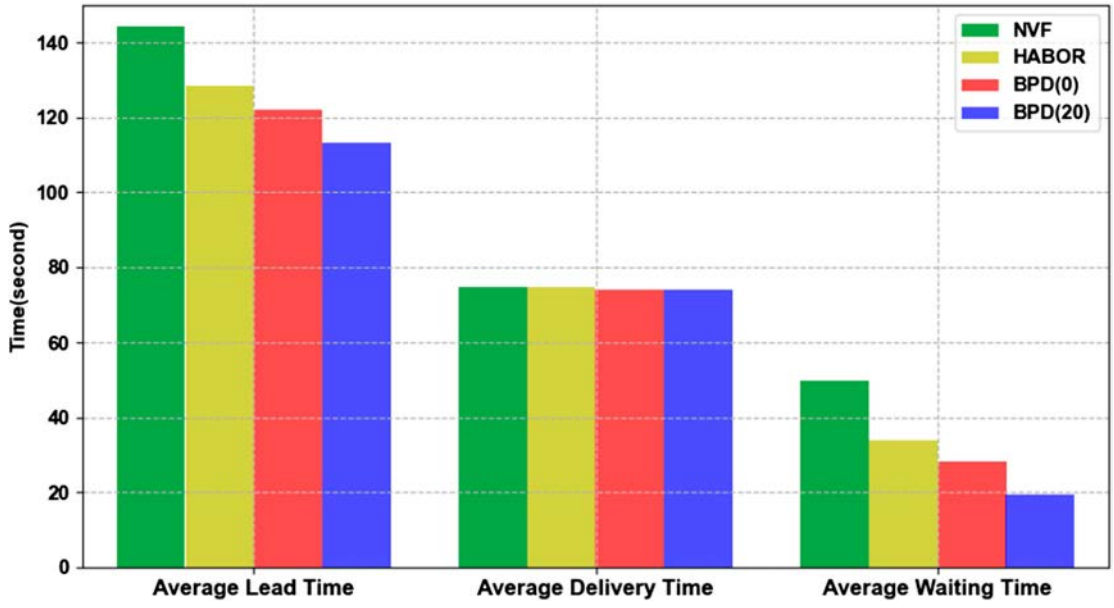


Figure 7. Performance of different dispatching methods.

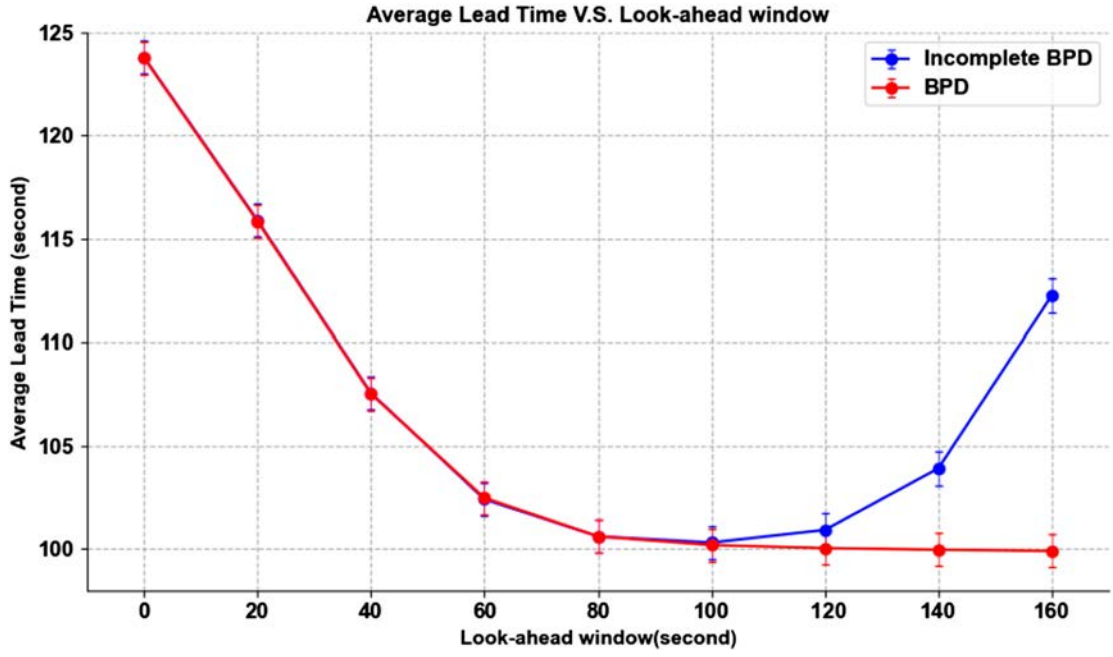


Figure 8. Performance of BPD with different look-ahead window.

future jobs, which indicates that the system does benefit from pre-allocating vehicles to the future jobs.

To further investigate the effect of the look-ahead window on the performance of BPD, we evaluate the BPD under different look-ahead window settings. The BPD just using Equation (7) as matching cost function and without any remedy, which we name as incomplete BPD, is also tested. The experiment result is displayed in Figure 8. When the look-ahead window is in the range of 0–100 s, both their average lead time declines as the look-ahead window increases. But the decrease gets slower. Incomplete BPD and BPD almost have the same performance at this range because the number of available jobs is still smaller than the number of vehicles. In this case, the incomplete BPD and BPD identically work well. However, as the look-ahead window continues increasing, more far future jobs will be taken as available jobs and the number of available jobs exceeds the number of vehicles. Then the incomplete BPD without any remedy falls into the issue we mentioned in Section 4. That is why the average lead time of incomplete BPD increases so rapidly. In contrast, the BPD still keeps good performance even

though the look-ahead window's increase no more brings performance improvement. This phenomenon is consistent with common sense that the information too far away actually does not help make dispatching decisions in the present.

5.3. Environment with prediction error

In practice, the job pre-arrival information provided by the MES may contain prediction errors. In the following subsections, we consider two virtual environments with prediction errors for the performance comparison of BPD and CWPD. We assume that an estimated job arrival list can be obtained from MES initially (Initial EAT_j prediction). As the manufacturing process progresses, the estimated job arrival list will be updated continuously to predict jobs' arrival times more reasonably (EAT_j update). The two virtual environments have different ways of EAT_j initialisation and update. It should be noted that the proposed BPD consistently shows better performance than NVF and HABOR even when the job arrival prediction is not entirely accurate; the following experiments are mainly to demonstrate the superiority of CWPD over BPD in the environment with prediction error.

5.3.1. Environment1: periodical arrival prediction

5.3.1.1. Initial EAT_j prediction. In the periodical arrival environment, the EAT_j of each job j is initialised according to the cycle time of the bay where job j is generated. By adding the arrival rate of jobs from bay m to other bays together, we can obtain the pooled arrival rate of jobs from the same bay m: $\lambda_m = \sum_{n=1}^{20} \lambda_{mn}$. It can be considered as the throughput of bay m, and the cycle time of bay m can be approximately calculated by $\frac{1}{\lambda_m}$. We assume that the arrival of jobs in a specific bay is strictly scheduled by its cycle time. That is to say, for the Nth job j generated from bay m, $EAT_j = N \times \frac{1}{\lambda_m}$.

5.3.1.2. EAT_j update. As aforementioned, the EAT_j of jobs needs to be updated to keep the prediction reasonable as the manufacturing process progresses. In this environment, a total of four kinds of update are involved.

(1) If the job generated at t_{now} is from bay m, then the EAT_j of Nth job that will be released at bay m will be rescheduled again by taking t_{now} as starting point, e.g. $EAT_j = t_{now} + N \times \frac{1}{\lambda_m}$
(2) For the job j that has already arrived, EAT_j should be updated to AAT_j, e.g. $EAT_j = AAT_j$
(3) For the job j who has not arrived yet but EAT_j is smaller than t_{now}, which means that the prediction has already turned out wrong. Thus EAT_j is updated by uniformly sampling from the interval between t_{now} and AAT_j, e.g. $EAT_j = U(t_{now}, AAT_j)$
(4) For the job j who has not arrived yet and EAT_j is also larger than t_{now} (within look-ahead window), EAT_j is update by uniformly sampling from the interval EAT_j and AAT_j (or AAT_j and EAT_j) w.p. $P_j = \frac{1}{k \cdot (AAT_j - t_{now}) + 1}$, where k is a constant

The reason for performing the fourth update is that the new information (i.e. the processing time of current operation of jobs) continuously becomes available, and this information helps to improve the prediction on the arrival time of future jobs. For a fixed k, the job whose AAT_j is closer to t_{now} has higher P_j because it is commonly accepted that the newly available information usually helps more on the prediction of nearer jobs.

We compare the performance of BPD and CWPD under different settings of k. In this experiment, the look-ahead window is set to 100 for both BPD and CWPD. We find that the prediction error is very high in this environment, meaning that the uncertainty on the prediction of future jobs' arrival time is also high. Thus the constant a and b in cf_j is set as 0 and 100, respectively. This setting ensures cf_j has a larger range to differentiate the jobs more appropriately. The experiment result is illustrated in Figure 9. The smaller k is, the more frequent EAT_j will be updated. Therefore, it can be observed that as the value of k increases, both the performance of BPD and CWPD decreases. However, CWPD can always give a satisfactory result than BPD, indicating that considering the certainty factor of prediction on jobs makes the proposed dispatching method more robust to the prediction error.

5.3.2. Environment2: initial prediction with Gaussian noise

5.3.2.1. Initial EAT_j prediction. In the Gaussian noise environment, the EAT_j is initialised by introducing a certain level of Gaussian noise to AAT_j for each job j. In other words, $EAT_j = N(AAT_j, \sigma)$, for each job j.

5.3.2.2. EAT_j update. EAT_j update in this environment is almost the same as the update in the periodic arrival environment except for two differences. One difference is that the first update in the periodic arrival environment is omitted because the prediction error in this environment is determined by σ instead of the cycle time of a specific port. The other difference is that the look-ahead window used in the above two environments is replaced by the number of future jobs looking ahead due to the uneven distribution of EAT_j resulted by the Gaussian noise.

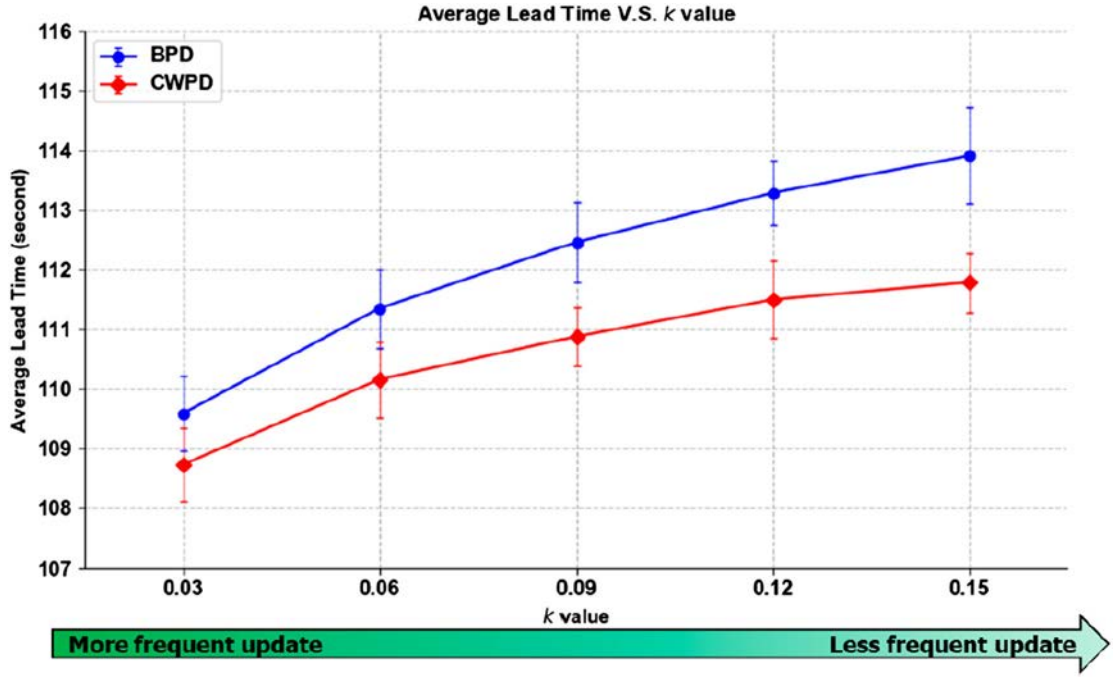


Figure 9. Performance comparison under different k settings.

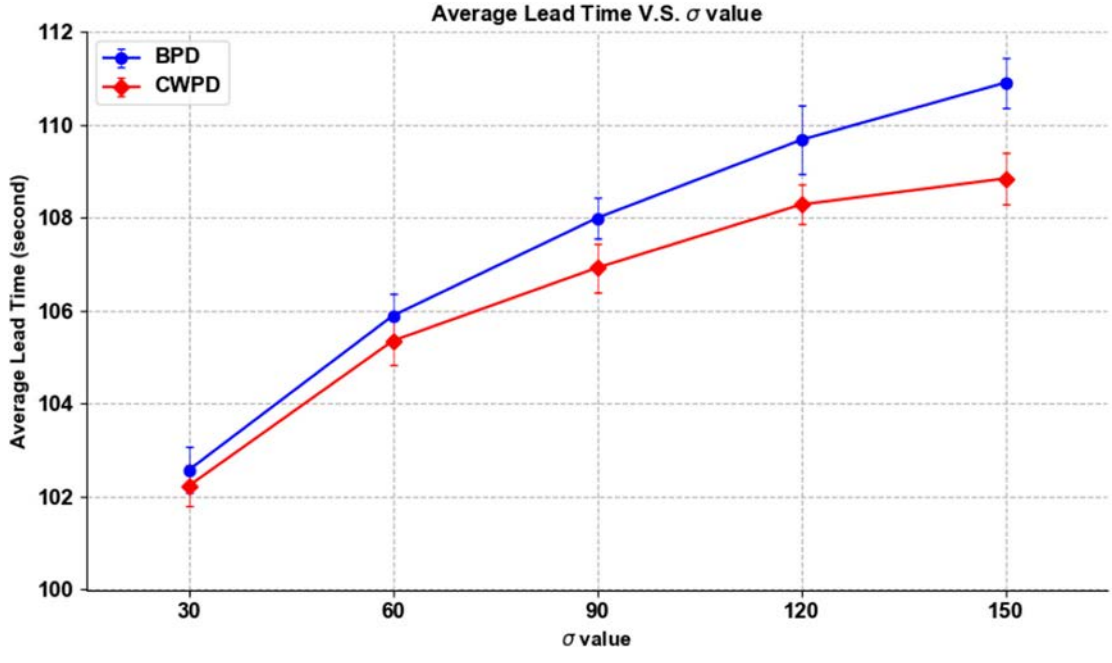


Figure 10. Performance comparison under different σ settings.

We contrast the performance of BPD and CWPD under different settings of σ. The number of jobs for looking ahead is set as 20, which is close to the number of jobs within the saturating look-ahead window in other environments considering the expected inter-arrival time of job arrivals is 6 s for the whole system; the constant k in P_j is set as 1; the constant b in cf_j is set as 100 while the constant a in cf_j is adaptively changed to σ. A special case is that if σ is 0, i.e. the environment becomes error-free, then we can set a to b. Figure 10 demonstrates the performance comparison of BPD and CWPD under different levels of Gaussian noise. We can easily observe that both the performance of BPD and CWPD worsen as the level of noise increases. However, CWPD consistently shows superior results than BPD, especially when the level of noise is high. The result again confirms the superiority

of CWPD over BPD in the environment with prediction error.

6. Conclusion

In this paper, we studied the predictive dispatching of vehicles by additionally considering the pre-arrival information of future jobs and the time needed for occupied vehicles to be idle. We verified the effectiveness of the proposed BPD and CWPD methods by conducting experiments on a sample semiconductor fab under different environment settings. The experiment results demonstrate that vehicle dispatching using the job state and the vehicle state expected in the near-future can significantly improve AMHS's performance in semiconductor fabs, which can mean a considerable cost saving for semiconductor manufacturers.

Moreover, considering the generality of the proposed methods, they might have the potential to be applied in other path-based material handling systems. The idea of predictive dispatching can also be applied by online ride-sharing companies such as Uber, DiDi.

Further studies may include the accurate prediction of the future transfer requests and vehicles' travel time because they can directly improve the performance of predictive dispatching. Utilising planned vehicle dispatching results to guide the routing of vehicles is also worth investigating.

Acknowledgments

We gratefully acknowledge the editor and anonymous reviewers for their valuable comments and suggestions, which have greatly helped improve the quality of this paper.

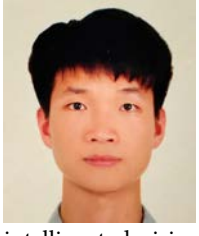
Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by SYNUS Tech [SYNUS Tech-KAIST-002].

Notes on contributors



Jiansong Wan received a B.S. degree in Logistics Management and Computer Science from Inha University in 2014, a M.S. degree in Industrial & System Engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 2020. He is currently pursuing a Ph.D. degree at KAIST. His research interest includes intelligent decision-making for logistics and manufacturing systems

Hayong Shin is a professor in the Department of Industrial & Systems Engineering at KAIST (Korea Advanced Institute of Science and Technology). Before joining KAIST, He worked for Chrysler Corp., CubicTek Co. and LG Electronics, developing commercial and in-house engineering software solutions. He received a BS from Seoul National University in 1985, an MS and a PhD from KAIST in 1987 and 1991, all in industrial engineering. His recent research interests are in the area of machine learning, simulation, geometric modelling for manufacturing system applications.. He can be reached at hyshin@kaist.ac.kr

References

Ahn, Kyuree, and Jinkyoo Park. 2021. "Cooperative Zone-Based Rebalancing of Idle Overhead Hoist Transportations Using Multi-Agent Reinforcement Learning with Graph Representation Learning." *IISE Transactions* 1–17. doi:10.1080/24725854.2020.1851823.

Aresi, Lucas, Stéphane Dauzère-Pérès, Claude Yugma, Moulaye Ndiaye, and Lionel Rullière. 2019. "AMHS Vehicle Management Policies in Semiconductor Manufacturing: A Short Review." In *2019 International Conference on Industrial Engineering and Systems Management (IESM)*, 1–6. IEEE.

Bartlett, Kelly, Junho Lee, Shabbir Ahmed, George Nemhauser, Joel Sokol, and Byungsoo Na. 2014. "Congestion-Aware Dynamic Routing in Automated Material Handling Systems." *Computers & Industrial Engineering* 70: 176–182.

Chaabane, Ahmed Ben, Stéphane Dauzère-Pérès, Claude Yugma, Lionel Rullière, and Gilles Lamiable. 2013. "Analyzing the Impact of Key Parameters of Vehicle Management Policies in a Unified AMHS." In *2013 Winter Simulations Conference (WSC)*, 3818–3828. IEEE.

de Koster, R. B. M., Tuan Le-Anh, and J. Robert van der Meer, and 2004. "Testing and Classifying Vehicle Dispatching Rules in Three Real-World Settings." *Journal of Operations Management* 22 (4): 369–386.

Gupta, Shreya, John J. Hasenbein, and Sanghyuk Park. 2020. "Improving Scheduling and Control of the OHTC Controller in Wafer Fab AMHS Systems." *Simulation Modelling Practice and Theory* 107: Article ID: 102190.

Hu, Hao, Xiaoliang Jia, Qixuan He, Shifeng Fu, and Kuo Liu. 2020. "Deep Reinforcement Learning Based AGVs Real-Time Scheduling with Mixed Rule for Flexible Shop Floor in Industry 4.0." *Computers & Industrial Engineering* 149: Article ID: 106749.

Huang, Chih-Wei, and James T. Lin. 2016. "A Pre-Dispatching Vehicle Method for a Diffusion Area in a 300 mm Wafer Fab." *Journal of Industrial and Production Engineering* 33 (8): 579–591.

Hwang, Illhoe, and Young Jae Jang. 2020. "Q (λ) Learning-Based Dynamic Route Guidance Algorithm for Overhead Hoist Transport Systems in Semiconductor Fabs." *International Journal of Production Research* 58 (4): 1199–1221.

Kim, Kap Hwan, and Jong Wook Bae. 2004. "A Look-Ahead Dispatching Method for Automated Guided Vehicles in Automated Port Container Terminals." *Transportation Science* 38 (2): 224–234.

Kim, Byung-In, Seungjin Oh, Jaejoon Shin, Mooyoung Jung, Junjae Chae, and Sujeong Lee. 2007. "Effectiveness of Vehicle

Reassignment in a Large-Scale Overhead Hoist Transport System." *International Journal of Production Research* 45 (4): 789–802.

Kim, Byung-In, Jaejoon Shin, Sangwon Jeong, and Jeongin Koo. 2009. "Effective Overhead Hoist Transport Dispatching Based on the Hungarian Algorithm for a Large Semiconductor FAB." *International Journal of Production Research* 47 (10): 2823–2834.

Kim, Junghoon, Gwangjae Yu, and Young Jae Jang. 2016. "Semiconductor FAB Layout Design Analysis with 300 mm FAB Data: 'Is Minimum Distance-Based Layout Design Best for Semiconductor FAB Design?'" *Computers & Industrial Engineering* 99: 330–346.

Kuo, Chung-Hsien, and Chien-Sheng Huang. 2006. "Dispatching of Overhead Hoist Vehicles in a Fab Intrabay Using a Multimission-Oriented Controller." *The International Journal of Advanced Manufacturing Technology* 27 (7–8): 824–832.

Lee, Sangmin, Junho Lee, and Byungsoo Na. 2018. "Practical Routing Algorithm Using a Congestion Monitoring System in Semiconductor Manufacturing." *IEEE Transactions on Semiconductor Manufacturing* 31 (4): 475–485.

Liao, Da-Yin, and Hsien-Sheng Fu. 2002. "Dynamic OHT Allocation and Dispatching in Large-Scaled 300 mm AMHS Management." In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, Vol. 4, 3630–3635. IEEE.

Liao, Da-Yin, and Chia-Nan Wang. 2006. "Differentiated Preemptive Dispatching for Automatic Materials Handling Services in 300 mm Semiconductor Foundry." *The International Journal of Advanced Manufacturing Technology* 29 (9–10): 890–896.

Qin, Wei, Jie Zhang, and Yinbin Sun. 2013. "Dynamic Dispatching for Interbay Material Handling by Using Modified Hungarian Algorithm and Fuzzy-Logic-Based Control." *The International Journal of Advanced Manufacturing Technology* 67 (1–4): 295–309.

Sarin, Subhash C., Amrusha Varadarajan, and Lixin Wang. 2011. "A Survey of Dispatching Rules for Operational Control in Wafer Fabrication." *Production Planning and Control* 22 (1): 4–24.

Schmaler, R., T. Schmidt, M. Schoeps, J. Luebke, R. Hupfer, and N. Schlaus. 2017. "Simulation Based Evaluation of Different Empty Vehicle Management Strategies with Considering Future Transport Jobs." In *2017 Winter Simulation Conference (WSC)*, 3576–3587.

Wang, Chia-Nan, and Li-Chin Chen. 2012. "The Heuristic Preemptive Dispatching Method of Material Transportation System in 300 mm Semiconductor Fabrication." *Journal of Intelligent Manufacturing* 23 (5): 2047–2056.

Appendices

Appendix 1

Suppose the available job set J has been sorted by the EAT_j, meaning that $RAT_1 = 0 < RAT_2 < RAT_3 < \cdots < RAT_{|J|}$. Next we prove the equivalence of the following two solutions.

(i) Limit the maximum number of available jobs to the number of vehicles $|V|$ while using $c_{jv} = (RT_{jv} + TTC_v - TTA_j)^+ = JWT_{jv}$.

(ii) Use $c_{jv} = JWT_{jv} + \alpha \cdot RAT_j$, where α is a very large number.

Proof: *Case 1*: $|J| \le |V|$.

The objective function of solution i is as follows.

$$\min_{x_{jv}} \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} c_{jv}x_{jv} = \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} JWT_{jv}x_{jv} \tag{A1}$$

The objective function of solution ii is as follows.

$$\begin{aligned}
&\min_{x_{jv}} \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} c_{jv}x_{jv} \\
&= \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} (JWT_{jv} + \alpha \cdot RAT_j)x_{jv} \\
&= \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} JWT_{jv}x_{jv} + \alpha \cdot \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} RAT_j x_{jv} \\
&= \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} JWT_{jv}x_{jv} + \alpha \cdot \sum_{j=1}^{|J|} RAT_j \underbrace{\sum_{v=1}^{|V|} x_{jv}}_{=1} \\
&= \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} JWT_{jv}x_{jv} + \underbrace{\alpha \sum_{j=1}^{|J|} RAT_j}_{\text{constant}} \\
&= \underbrace{\sum_{j=1}^{|J|}\sum_{v=1}^{|V|} JWT_{jv}x_{jv}}_{=\,(A1)}
\end{aligned} \tag{A2}$$

Case 2: $|J| > |V|$

In this case, because solution i limits the number of available jobs to $|V|$. Thus $|J| = |V|$ in solution i. The objective function of solution i is as follows.

$$\min_{x_{jv}} \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} c_{jv}x_{jv} = \sum_{j=1}^{|V|}\sum_{v=1}^{|V|} JWT_{jv}x_{jv} \tag{A3}$$

The objective function of solution ii is as follows.

$$\begin{aligned}
\min_{x_{jv}} \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} c_{jv}x_{jv} &= \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} (JWT_{jv} + \alpha \cdot RAT_j)x_{jv} \\
&= \sum_{j=1}^{|V|}\sum_{v=1}^{|V|} (JWT_{jv} + \alpha \cdot RAT_j)x_{jv} \\
&\quad + \sum_{k=|V|+1}^{|J|}\sum_{v=1}^{|V|} (JWT_{kv} + \alpha \cdot RAT_k)x_{kv}
\end{aligned} \tag{A4}$$

Because $\sum_{j=1}^{|J|} x_{jv} = 1$, thus in (A4), $x_{jv} + x_{kv} = 1$, where both x_{jv} and x_{kv} are binary. Next we prove that to minimise (A4), x_{kv} must all be 0 for $k = |V| + 1, |V| + 2, \ldots, |J|$.

If $x_{kv} = 0, \forall\, k = |V|+1, |V|+2, \ldots, |J|$, total dispatching cost is:

$$\begin{aligned} TC_1 &= \sum_{j=1}^{|V|}\sum_{v=1}^{|V|}(JWT_{jv} + \alpha \cdot RAT_j)x_{jv} \\ &\quad + \underbrace{\sum_{k=|V|+1}^{|J|}\sum_{v=1}^{|V|}(JWT_{kv} + \alpha \cdot RAT_k)x_{kv}}_{=0} \\ &= \sum_{j=1}^{|V|}\sum_{v=1}^{|V|}(JWT_{jv} + \alpha \cdot RAT_j)x_{jv} \end{aligned}$$

Now suppose one of x_{kv} becomes 1,e.g. $x_{lv} = 1, |V|+1 \le l \le |J|$, then one of x_{jv} should become 0, e.g. $x_{iv} = 0, 1 \le i \le |V|$. The total dispatching cost in this case is:

$$TC_2 = \sum_{j=1, j\neq i}^{|V|}\sum_{v=1}^{|V|}(JWT_{jv} + \alpha \cdot RAT_j)x_{jv} + (JWT_{lv} + \alpha \cdot RAT_l)$$

Then we have:

$$\begin{aligned} TC_2 - TC_1 &= \sum_{j=1, j\neq i}^{|V|}\sum_{v=1}^{|V|}(JWT_{jv} \\ &\quad + \alpha \cdot RAT_j)x_{jv} + (JWT_{lv} + \alpha \cdot RAT_l) \\ &\quad - \sum_{j=1}^{|V|}\sum_{v=1}^{|V|}(JWT_{jv} + \alpha \cdot RAT_j)x_{jv} \\ &= (JWT_{lv} + \alpha \cdot RAT_l) - (JWT_{iv} + \alpha \cdot RAT_i) \\ &= \alpha(RAT_l - RAT_i) + (JWT_{lv} - JWT_{iv}) \end{aligned}$$

Because $RAT_l - RAT_i > 0$ and α is a dominantly large number, thus no matter how $(JWT_{lv} - JWT_{iv})$ will be, $TC_2 - TC_1 > 0$. This indicates that any change of x_{kv} from 0 to 1 will increases the dispatching cost. Hence, to minimise the total dispatching cost, all x_{kv} should be 0.

Now we continue our proof from (A4). Because all x_{kv} should be all 0 in (A4), then the objective function can be rewritten as:

$$\begin{aligned} \min_{x_{jv}} \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} c_{jv}x_{jv} &= \sum_{j=1}^{|J|}\sum_{v=1}^{|V|}(JWT_{jv} + \alpha \cdot RAT_j)x_{jv} \\ &= \sum_{j=1}^{|V|}\sum_{v=1}^{|V|}(JWT_{jv} + \alpha \cdot RAT_j)x_{jv} \\ &\quad + \underbrace{\sum_{k=|V|+1}^{|J|}\sum_{v=1}^{|V|}(JWT_{kv} + \alpha \cdot RAT_k)x_{kv}}_{=0} \\ &= \sum_{j=1}^{|V|}\sum_{v=1}^{|V|}(JWT_{jv} + \alpha \cdot RAT_j)x_{jv} \\ &= \underbrace{\sum_{j=1}^{|V|}\sum_{v=1}^{|V|} JWT_{jv}x_{jv}}_{=(A3)} \end{aligned} \tag{A5}$$

The last step uses the conclusion we have drawn for the case $|J| \le |V|$. Finally, we complete our proof. ■

Appendix 2

A.1

The CWPD version of the two solutions discussed in Appendix 1 is as follows.

(i) Limit the maximum number of available jobs to the number of vehicles $|V|$ while using $c_{jv} = (RT_{jv} + TTC_v - TTA_j)^+ \times cf_j = JWT_{jv} \times cf_j$.
(ii) Use $c_{jv} = JWT_{jv} \times cf_j + \alpha \cdot RAT_j$, where α is a very large number.

By following the same reasoning process of Appendix 1, we can also prove that the above two approaches are equivalently effective for the CWPD scenario. Hence, we select the first approach as an example for the following discussion.

Without multiplying cf_j, the total dispatching cost is:

$$\min_{x_{jv}} \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} c_{jv}x_{jv} = \sum_{j=1}^{\min\{|J|,|V|\}}\sum_{v=1}^{|V|} JWT_{jv}x_{jv} \tag{A6}$$

By multiplying cf_j, the total dispatching cost is:

$$\min_{x_{jv}} \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} c_{jv}x_{jv} = \sum_{j=1}^{\min\{|J|,|V|\}}\sum_{v=1}^{|V|} JWT_{jv}cf_jx_{jv} \tag{A7}$$

We can see that the original objective changes into finding x_{jv} so that the sum of cf_jJWT_{jv} can be minimised. For the job j with larger cf_j, a small increase in JWT_{jv} may increase a large portion in total dispatching cost. Therefore, to minimise the total cost, the job j with larger cf_j is more likely to be matched with vehicle v that minimises its waiting time JWT_{jv}.

A.2

If the b and a in the definition of cf_j are the same, then $cf_j = b$ for all the jobs. Therefore, (A7) can be rewritten as:

$$\begin{aligned} \min_{x_{jv}} \sum_{j=1}^{|J|}\sum_{v=1}^{|V|} c_{jv}x_{jv} &= \sum_{j=1}^{\min\{|J|,|V|\}}\sum_{v=1}^{|V|} JWT_{jv}bx_{jv} \\ &= \underbrace{b}_{\text{constant}} \times \sum_{j=1}^{\min\{|J|,|V|\}}\sum_{v=1}^{|V|} JWT_{jv}x_{jv} \\ &= \underbrace{\sum_{j=1}^{\min\{|J|,|V|\}}\sum_{v=1}^{|V|} JWT_{jv}x_{jv}}_{=(B1)} \end{aligned}$$